UNIVERSIDAD SAN FRANCISCO DE QUITO

Colegio de Posgrados

Development of a tool for determining speed and road directions using GPS vehicle devices and free resources

José Luis Aragón Osejo

Richard Resl, Ph.Dc., Director de Tesis

Tesis de grado presentada como requisito
para la obtención del título de Magister en Sistemas de Información Geográfica

Quito, enero de 2014

Universidad San Francisco de Quito

Colegio de Posgrados

HOJA DE APROBACIÓN DE TESIS

Development of a tool for determining speed and road directions using GPS vehicle devices and free resources

José Luis Aragón Osejo

Richard Resl, Ph.Dc. ..……..……
Director de Tesis

Pablo Cabrera, Ms. ……..
Miembro del Comité de Tesis

Richard Resl, Ph.Dc. ……..
Director de la Maestría en Sistemas de Información Geográfica

Stella de la Torre, Ph.D. ……..
Decana del Colegio de Ciencias Biológicas y Ambientales

Víctor Viteri Breedy, Ph.D. ……..
Decano del Colegio de Posgrados

Quito, enero de 2014

Firma:

Nombre: JOSÉ LUIS ARAGÓN OSEJO

C. I.: 1708885643

Quito, 10 de enero 2014

Agradecimientos y dedicatoria

“El presente trabajo es un esfuerzo de aprendizaje, que espero rinda frutos concretos en el futuro, siguiendo los principios que me guían.

Por eso, lo dedico a quienes me han apoyado en el camino. Especialmente a mi familia, nunca dejaré de crecer gracias a ellos.”

Resumen

Los dispositivos de localización satelital (GPS) en vehículos son cada vez más populares en países como el Ecuador, lo que significa que datos GPS se generan todos los días (por ejemplo los datos provistos para este trabajo, provenientes de una flota de 50 buses de la ciudad de Portoviejo, Ecuador). Estos datos pueden luego convertirse, si son procesados, en información relacionada con redes viales.

Aunque estos datos estén disponibles como archivos de texto, para adquirir información útil desde ellos debe existir una herramienta desarrollada con este propósito, que sea accesible para usuarios comunes, usando recursos libres, y capaz de procesar y proveer como resultado una red vial complementada y utilizable.

Este trabajo intenta proveer una herramienta para realizar este procesamiento, usando recursos libres, como Python, Quantum GIS (QGIS) y OpenStreetMap (OSM). Se desarrolló un algoritmo para adquirir información sobre direcciones de vías y promedios de velocidad, usando los datos disponibles de GPS de vehículos. Para esto, se modificó un complemento de QGIS, mediante las capacidades de geoprocesamiento de Python. Se diseñó e implementó una fase de pruebas para comprobar el funcionamiento de los algoritmos y de la capacidad del hardware, especialmente trabajando con archivos grandes.

Durante esta fase de pruebas surgieron errores que obligaron a realizar las correcciones respectivas hasta que el procesamiento se desarrolló de forma correcta. Los resultados obtenidos demostraron que se puede adquirir información de sentidos viales y promedios de velocidad, a pesar de los eventos que causan confusión en la herramienta desarrollada.

Abstract

Vehicle GPS devices in countries like Ecuador have increased, meaning that new GPS data is collected every day (for example the data provided for this work, collected from a 50 buses fleet in the city of Portoviejo, Ecuador). This data can later become (if processed) information corresponding to road networks.

Although GPS data is available as text files, to acquire useful information from it there must exist a tool developed for this matter, accessible to common users (using free resources), and able to process it and provide a usable completed road network as a result.

This work intends to provide a tool to perform this processing, using free resources, such as Python, Quantum GIS (QGIS) and OpenStreetMap (OSM). An algorithm was developed to acquire road direction and speed average information, using provided GPS data from vehicles. For this, OSM plugin from QGIS was modified, through Python geoprocessing capabilities. A testing phase was designed and implemented to try out algorithms performance, and hardware capacity, specially handling large files.

During this testing phase programming errors appeared obligating to correct these issues until processing was done correctly. Results demonstrated that road direction and speed average information can be acquired, despite events that can confuse the tool.

Tabla de contenidos

Lista de Tablas

Lista de Figuras

1. Introduction

Tracking and routing are becoming very common scheduling components for vehicle, bicycle or any other transport means, when people or institutions deal with route planning. In 2009, about half of United States transit buses were using GPS devices for tracking purposes (American Public Transportation Association, 2009).

Schedules can take into account external conditions such as traffic, vehicle size, cargo, hour restrictions, weekend days, and others, although, most times it is just a plain route to reach a target site.

Nowadays there are several online and desktop tools that can achieve routes in a very trustable and fast way. There are also several algorithms used for routing, implemented through software development continuously improved, making its use intuitive so not only GIS specialists can profit from it. Most of these tools are now widely available through Internet.

These tools need good local cartographic information to achieve results, but that's not always the case, or it is not easily available.

1.1. Background

Needs for monitoring, surveillance, control were the reasons to develop a wide range of GPS devices meant to be used in cars, buses, motorcycles and others. These devices collect current data, as vehicles travel their day to day routine paths through city streets. This data is exceptionally used and analyzed, but most times it is not even stored.

The devices send the GPS location as latitude and longitude coordinates to a server, through GPRS (cellular signal), satellite signal, or other means. The server recognizes this data, and might use it to perform processing analysis, such as calculating speed information, vehicle productivity, and distance to a given facility.

GPS devices are used for routing purposes, which means to find the best way to go from a starting point to another point or points. Algorithms used to resolve this matter might take into account many elements like road restrictions, ground elevation and others, according to the complexity of the problem and the resources available, and especially road directions.

Nowadays GPS devices are very common, but depending on legal contracts, there is restricted access to this data and no storage involved. Vehicle owners cannot process or do not care about these data, as long as the service they requested is provided. So a great amount of data collected is never used.

On one hand, before GIS desktop software and open source software came into scene, data processing needed a GIS specialist (or a GIS department), to handle difficult and expensive software. On the other hand, the cartographic information needed is generated through expensive field and desktop work. Companies investing in the generation of this cartographic information charge high prices to make profit, otherwise this kind of investment cannot be done. This causes that although the information and software needed to do routing exists, not all companies or institutions can afford them, so the implementation of GIS analysis and custom processing is not possible in most cases, with the exception of few large institutions.

1.2. Problem

Nowadays business are conceived and designed to be efficient, in terms of money and time. Resources are handled through sharp flows intended to enhance productivity with minimum expense. Even common drivers are usually trying to spend minimum time on the road.

In the case of routing, vehicles have to attend their target sites according to previous planning, with minimum delay. This includes the possibility that a vehicle gets stolen or damaged. Besides that, as unforeseen events happen, real time rescheduling could be needed. Therefore vehicles are tracked, so their productivity is assured.

Currently online free systems like Google Maps allow users for tracking purposes, but only for a small amount of cars. A paid license is necessary beyond that.

Some countries started the development of free maps such as OpenStreetMap (OSM) many years ago, and have now consolidated free maps, available for anyone to use them. Some commercial companies, which used Google Maps before, now prefer OSM because of its reliability and because of its license permissions.

Road networks of cities should be updated as often as possible so routing results are accurate and useful, because streets get closed, their directions get changed, and as cities grow more roads are added to the street network. Updates can be very expensive and usually depend on private offer.

Conversely, free resources imply no cost, but depend on their developers and the community involved to be up to date. Specific technical knowledge is needed to assemble free resources and develop a specific tool like the one this work is trying to obtain. This is not always available, as developers are not capable to do all the software development needed. Also, satisfying documentation and adequate support are not always available.

Currently in Ecuador, there are various sources of updated cartographic information but they are not free, and can be used only through the software provided by the vendor, this means to be used only the way the provider intended. Free maps are neither updated nor complete.

1.3. Hypothesis

Vehicle GPS data can be used to obtain road directions and speed information to complement a road network, using only free resources.

Determination of road directions and speed depends on the data provided, its format, technical characteristics, and quantity. If the combination of these aspects does not achieve a solid base to determine road directions and speed, no algorithm, program or code can get trustable results.

A logical procedure has to be developed, taking into account essential routing procedures, and use road directions and speed information so the information incorporated does not stop, interfere or mislead routing calculating processes.

Free resources are going to be used, this implies that there’s not going to be a technical service to demand for an answer or solution to any given problem,

instead programmers and user communities might provide a solution. There is technical information, especially online, but it is not always updated or it is incomplete.

The solution to these problems is a program that works with a set of GPS data provided; it must calculate speed, as well as determine road directions. If possible, the result has to be a file ready to be used. Users could use the program regularly, so it must be as easy and intuitive as possible. Also there might be the need to use it with different type of GPS data or including other functionalities, so it must allow expansion in the future.

1.4. Research question

How and to what extent vehicle GPS data can be used to obtain road directions and speed information, using free resources?

To answer the proposed research question, a logical flow is exposed and taken into account, from income to output. From this flow, results are obtained, and demonstrate the capabilities of the program and to what extent the question has been answered, and provide (or not) a satisfying solution, to be discussed in terms of scope and limitations.

1.5. Targets of this work

This work assembles all of the above mentioned and tries to improve the current free resources situation for routing technology in Ecuador, using free resources.

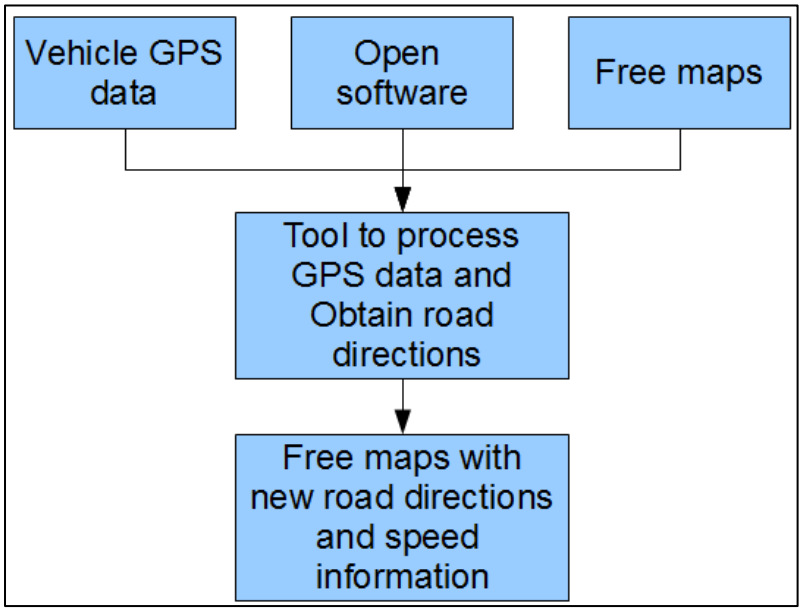


Figure 1.- Basic Project Flow

General objective

- Develop a tool for determining speed and road directions using GPS vehicle devices and free resources

Specific objectives

- Find a convenient methodology to process and determine speed and road directions
- Establish free resources to develop the tool
- Write code to customize the tool
- Achieve a successful run of the entire process

1.6. Definitions of terms

Some specific terms are repeatedly used along this work, outside GIS environment, so they are defined here.

- GPRS: General Packet Radio Service (cellular communication system)
- GPS: stands for Global Positioning System
- GPS device: apparatus used to catch GPS satellites signal and calculate a location
- Road direction: sense in which a vehicle is allowed to go over a road
- Speed average: speed calculated from a set of determined speeds, taking into account high and low speed values
- Speed legal limit: legal limit according to the type of road

1.7. Presumptions of the author

It was expected that the information provided containing GPS data was adequate regarding its precision and format to be processed.

Hardware resources are the same as regular GIS specialist could get, that is a desktop computer with regular processing capabilities.

Open software (and proprietary) is not always easily compatible with other resources; the tool that was assembled was not tested on all possible platforms, or all possible software versions, including later versions.

Some open resources (e.g. OpenStreetMap) are maintained by a large community, with both local and worldwide organization voluntarily engaged. There might be some situations or procedures with no literature to relay on.

1.8. Assumptions of this thesis study

The product expected from the tool to be created is a dataset ready to be used, which includes new road directions and speed information. This depends on the initial data and the inherent capabilities of the resources to be used.

The tool itself intends to be friendly and straightforward, so its use is encouraged. Also, the tool should be available under the correct license terms to encourage common users and possible feedback.

2. Literature Review

2.1. Open resources

A creator decides how his or her creation may be made available to the public, or if it is not available at all. As he claims his or her rights, he may also decide how the creation is available to others, conditions put into license documents. Licenses claim intellectual property, establishing “the legal rights which result from intellectual activity in the industrial, scientific, literary and artistic fields” (World Intellectual Property Organization, 2004). Otherwise, the creation goes to the public domain.

Nowadays, regarding software and other resources, there are different license options: open source, free and commercial licenses, with many inner variations, like dual licenses agreements (both commercial and non-commercial).

Depending on the importance given to a license issue, some open source licenses are incompatible, or compatible. Besides, intellectual property rights are limited territorially; they exist and can be exercised only within the jurisdiction of a country (World Intellectual Property Organization, 2004).

In this work’s context and the topics dealt in it (software and geographic information), open resources refer to any resource that can be used without a fee and whose internal structure may be known to anyone (i.e. software’s code, vector map files, etc.).

“Open Source” implies three dimensions as it applies to computing (Tomer, 2002):

1. A philosophy about computing and sharing programming code to improve the quality of computing.
2. A wide array of operating systems and applications that have been developed under this philosophy.
3. A general approach to the treatment of intellectual property, usually in reference to licensing software or documentation.

It started as a reaction to software restrictions that hardware sellers imposed. As the open source movement was growing, some dedicated organizations appeared, like the Free Software Foundation or the Open Source Initiative. The latter proposes the following guidelines (among others) to define open source licenses (Open Source Initiative, n.d.):

1. Free Redistribution: no fee required
2. Source Code: access to code to modify it
3. Derived Works: must be allowed and available to distribution
4. License Must Not Be Specific to a Product
5. License Must Be Technology-Neutral: not predicated on any individual technology or style of interface.

General Public License (GPL) is one of the most popular open source licenses, and follows these guidelines. Other licenses following GPL guidelines are known as being “GPL Compatibles”. “GPL Compatibility” means, among other issues, that any derivative work must itself be distributed under GPL, and no additional restrictions may be placed on the redistribution of either the original

work or a derivative work (Fogel, 2005). The requirement for derivative work to use the same license as the original work (also known as reciprocal or viral) is known as “Copyleft”.

2.2. OpenStreetMap

2.2.1. User generated content

There are several initiatives that allow users to contribute to a database, making its content available to others, which are called “wikis”. “Wiki is a website that lets people freely create, edit, and link a collection of articles. (...) Wikis allow the content and the structure to be changed by a community” (Barret, 2008).

Wikipedia, for example, shares its articles freely as the community involved improves its contents. Wikipedia has quickly become one of the largest and most accessed Web sites in the Internet, appearing atop GoogleTM searches for millions queries (Chatfield, 2009).

Wiki concept is used by OpenStreetMap.

2.2.2. OpenStreetMap overview

OpenStreetMap (OSM) main web site and documentation are mainly available through http://www.openstreetmap.org . Its aim is to eventually have a record of every geographic feature on the planet in a free geographic database of the world.

OSM is built through volunteer effort. Each contributor develops a map of his local streets using GPS tracking; and individual contributions are assembled and reconciled into a single patchwork. Extensive metadata is incorporated, since

each piece of the patchwork may have different levels of accuracy and may have been acquired at different dates (Goodchild, 2007).

Nowadays there are over a million users (MapBox, 2013), and most big and medium cities have already been fully mapped.

2.2.3. OSM data

OSM data is stored in its own format, which enables some features and capabilities. These specifications are explained in the next chapters.

2.2.3.1. OSM: XML based format

Extensible Markup Language (XML) is a simple, flexible text format. It was originally conceived to help with large-scale electronic publishing; although is now also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere (World Wide Web Consortium, 2012).

OSM uses a format based on XML structure to store spatial data, which was specially developed. It represents geo-spatial data “in the form of nodes (single points), ways (sequence of points that define a line), areas (closed ways that represent polygons), and relations (collections of other elements) (Westra, 2010).

An example of OSM code is explained below, which was downloaded from OSM website, and edited to fit the explanation.

General structure starts with a header and general data specifications, like XML and OSM data version, license and bounds. The part containing nodes,

ways and relations may be very long, depending on the spatial extent and the amount of data within that extent.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.1.0" copyright="OpenStreetMap and contributors"
attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/">
 <bounds minlat="-0.1561420" minlon="-78.4777500" maxlat="-0.1531750" maxlon="-78.4760230"/>
<!-- Nodes, ways and relations. -->
</osm>
```

A node is defined by its coordinates (latitude and longitude) and its identification number, which is unique for that node worldwide. It can show other useful information, like changeset identification, which can be used to remove entire sets of information if a problem is detected.

```
<node id="269572065" lat="-0.1549247" lon="-78.4768789" user="rafatenor" uid="118380"
visible="true" version="11" changeset="5078579" timestamp="2010-06-26T06:27:22Z"/>
```

Nodes might have tags, providing extra information about it. Tags are already defined (and are continually getting improved) so all OSM users can profit from that information. OSM uses “key – value” format to define a node, for example k=”highway” and v=”bus_stop” defines a bus stop.

```
<node id="262203745" lat="-0.1589536" lon="-78.4837410" user="LLAQWA" uid="77114"
visible="true" version="15" changeset="10971169" timestamp="2012-03-13T19:59:29Z">
  <tag k="highway" v="bus_stop"/>
  <tag k="public_transport" v="stop_position"/>
 </node>
```

Ways are collections of nodes, so a list of nodes is given. Ways might also have tags, different from node tags as they show ways information. If a way is closed, which means that its first and last node are the same, then that way is a polygon, and might add some other specific tags.

```
<way id="24812835" user="lxbarth" uid="589596" visible="true" version="7" changeset="15915658"
timestamp="2013-04-30T02:08:35Z">
<nd ref="269572115"/>
<nd ref="269572240"/>
<tag k="hgv" v="no"/>
<tag k="highway" v="residential"/>
<tag k="name" v="El Morlan"/>
</way>
```

Relations are collections of nodes, ways and other relations. They are used when different features define spatial information, for example, a forbidden left turn, a compulsory turn right, and others, which involve two or more ways and intersection nodes.

```
<relation id="2081539" user="LLAQWA" uid="77114" visible="true" version="1"
changeset="10969244" timestamp="2012-03-13T17:16:50Z">
<member type="relation" ref="2081508" role=""/>
<member type="relation" ref="88867" role=""/>
<tag k="bus" v="regular"/>
<tag k="fixme" v="yes"/>
<tag k="name" v="Q5 Comité de Pueblo"/>
<tag k="network" v="Metrobus-Q"/>
<tag k="on_demand" v="no"/>
<tag k="type" v="route_master"/>
</relation>
```

2.2.3.2. OSM tags

As mentioned before, tags used in OSM code are already defined, although related discussion and improvement is permanent. A lot of them are common to all users in a given language; others are specific to a country.

Regarding Ecuador defined tags; they have not been translated yet from original English tags. Values to “highway” tag are: Motorway, Trunk, Primary, Secondary, Tertiary, Residential, living_street, Unclassified, Road, and Pedestrian (OpenStreetMap, 2011).

The other tags used for this study are shown below (OpenStreetMap, 2013).

- Directions are defined by “oneway” tag, and the values are “yes” (oneway from starting node to ending node), “-1” (from ending node to starting node), and “no” (two way street).

- The tag for maximum speed is “maxspeed”, and the value for Ecuador is 50 km/h (Asamblea Nacional Constituyente - Ecuador, 2012).
- There is no average speed tag, so the tag used is “avgspeed”.

2.2.4. License

OSM uses Open Database License (ODbL) 1.0. This license agreement was first used by OSM, and it guides the usage of its spatial data contents (OpenStreetMap, 2011).

ODbL allows users to (Open Data Commons, n.d.):

- Share: To copy, distribute and use the database.
- Create: To produce works from the database.
- Adapt: To modify, transform and build upon the database.

These rights are granted as long as users:

- Attribute any public use of the database, or works produced from the database. It must make clear to others the license of the database.
- Share-Alike, any adapted database must also be offered under the ODbL.
- Keep open, any distribution is permitted, as long a version without restrictions is also redistribute.

The cartography in OSM map tiles and the documentation are licensed under the Creative Commons Attribution-ShareAlike 2.0 license (OpenStreetMap, n.d.).

2.3. Open Geographic Information Systems

Open source GIS programs are based on different base programming languages. There are three main groups of open source GIS (outside of web GIS) in terms of programming languages: “C” languages, Java, and .NET (GISLounge, 2012).

The first group is the more mature one, for it has been working the longest and has a long history of reuse of code. Popular “C” based GIS software includes GRASS and QGIS. The second group uses JAVA; GeoTools, Geoserve, and OpenMap, are among the most popular in this group. The third most influential group integrates applications that use “.NET**”**, like SharpMap and WorldWind.

2.3.1. QGIS overview

The Quantum GIS project was officially born in May of 2002, when coding began. The first release supported only PostGIS layers (The Quantum GIS project, n.d.). There are at least 100,000 QGIS users around the world (Sherman, 2011). Currently, version 2.0.1 is available.

QGIS project offers QGIS Desktop, QGIS Browser, QGIS Server and QGIS Client. The desktop application used for this work is version 1.7.0 (Wroclaw), and its principal features include (Quantum GIS web site, n.d.):

- Direct viewing of vector and raster data in different formats and projections. Supported formats include PostGIS and ESRI shapefiles, also raster formats like GRASS locations and mapsets, and online spatial data (WMS , WMS-C (Tile cache), WFS and WFS-T).

- Mapping and interactive exploration of spatial data. Tools include on-the-fly reprojection, print composer, edit/view/search attributes, and others.
- Creating, editing and exporting spatial data using digitizing tools for vector features, field and raster calculator.

2.3.2. Geoprocessing with Python (plugins)

QGIS has been designed with plugin architecture. This allows many new features/functions to be easily added to the application. There exist two kinds of plugins (Open Source Geospatial Foundation, 2011):

- Core Plugins, which are maintained by the QGIS Development Team and are automatically part of every QGIS distribution. They are written in C++ or Python.
- External Plugins, which are currently all written in Python. They are stored in external repositories and maintained by the individual authors.

OpenStreetMap plugin is a Python plugin.

2.3.3. License

QGIS is licensed under the GNU General Public License. OpenStreetMap plugin code is available and can be modified also under the GNU GPL license terms.

2.4. Python

2.4.1. Python overview

Python is an open-source general-purpose programming language. Python has distributions available for many platforms, like Microsoft Windows, Apple Mac OS X, GNU/Linux (Wikibooks, 2013). This makes the programs very portable, as any program written for one platform can be used in another.

Big companies or institutions are using Python nowadays. For example NASA has used Python for its software systems and has adopted it as the standard scripting language for its Integrated Planning System. Python is also extensively used by Google to implement many components of its Web Crawler and Search Engine & Yahoo! for managing its discussion groups (Wikibooks, 2013).

According to its web site, Python has some distinguishing features (Python Software Foundation, 2013):

- very clear, readable syntax
- strong introspection capabilities
- intuitive object orientation
- natural expression of procedural code
- full modularity, supporting hierarchical packages
- very high level dynamic data types
- extensive standard libraries and third party modules for virtually every task
- extensions and modules easily written in C, C++

- embeddable within applications as a scripting interface

Currently, there are two major Python stable distributions in Python web site (http://www.python.org/), versions 2.7.6 and 3.3.3, that are the most updated versions of Python 2.x and 3.x releases. These two releases are not compatible, so a lot of libraries still use 2.x versions of Python. That is the case of QGIS 1.7.0, which uses version 2.5.

2.4.1. Python programming

Python is a high level programming language, and it includes several libraries and modules which functionality is exploited through Python classes.

Functions and classes are used so excerpts of code can be easily invoked without having to rewrite them. Instead functions or classes are called. An example of a class and inner functions is shown next.

```
class MyClass:
        def __init__(self, parameter1, parameter2):
                #do something when class is called
        def  function1(self, parameter3, parameter4):
                #do something when function1 is called
                return value1
        def  function2(self, parameter5, parameter6):
                #do something when function2 is called
```

Functions __init__ are already defined by Python to be used when a class is called, at starting point. After this, functions can be called after creating an instance of that class. Parameter “self” refers to the new instance itself.

```
instance = class(parameter1, parameter2)
instanceValue1 = instance.function1(parameter3,parameter4)
instance.function2(parameter5,parameter6)
```

As for programming itself, some conventions have been defined so code can be easily readable and modified. Appendix 2 shows the most important conventions used.

2.4.2. Python and QGIS

Python has become one of the key languages to use with GIS. This is mainly because it's very commonly available and integrates well with the C++ code which forms the basis of a lot of GIS functionality (MacWright, 2012).

Several open software GIS applications use Python, among the most important are GDAL/OGR Python bindings, shapely, pyproj, Fiona, geopy, geodjango, MapFish, owslib, MapServer, Mapnik (Corti, 2012).

QGIS has Python language scripting capabilities, to enable usage of QGIS libraries, through PyQGIS bindings. (Dobias, 2011). It uses Python 2.5.

2.4.3. License

Python distribution uses Python Software Foundation License, administered by Python Software Foundation and compatible with GPL license. This open source license makes it freely usable and distributable, even for commercial use (Python Software Foundation, 2013).

2.5. Qt4

2.5.1. Qt4 overview

Qt is a platform development framework, designed to create applications and user interfaces. It supplies APIs for C++ and CSS/JavaScript-like

programming, an integrated development environment, including UI designer tools and debugging. It is usable with all major platforms, like Windows and Linux (Digia, 2013).

Some of the applications developed with Qt are KDE, Opera, Google Earth, Skype, VLC, Maya or Mathematica (ZetCode, 2012), also QGIS (The Quantum GIS project, n.d.).

Qt has bindings allowing programmers to use it with several programming languages. There is the Python binding, PyQt. The version used for this study is PyQt4, and it is composed by many modules, like QtCore which contains non Graphical User Interface (GUI) classes, and QtGui module which contains GUI widgets.

2.5.2. License

Qt4 and PyQt4 have open and commercial licenses, although not the same ones and they must be compatible. Both commercial licenses are used for creating commercial applications. Qt4 is licensed under LGPL v. 2.1 (Qt Project Hosting, 2013). PyQt4 is licensed on all platforms under a commercial license, the GPL v2 and the GPL v3. PyQt4, unlike Qt, is not available under the LGPL (Riverbank Computing Limited, 2011).

PyQt used under GPL is compatible with Qt under LGPL and GPL licenses. The commercial version of PyQt is compatible with both the commercial and LGPL versions of Qt (Riverbank Computing Limited, 2013).

2.6. Used open resources summary

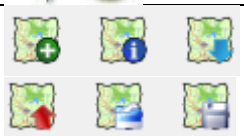
Resource	Logo	Version	License
QGIS		1.7.0 Wroclaw	GPL
OpenStreetMap plugin		0.5	GPL
Python		2.5	Python License
Qt4		4.7.1	LGPL v. 2.1
OpenStreetMap			ODbL 1.0.

Table 1.- Resources used

2.7. Vehicle tracking

2.7.1. Software and hardware

Vehicle tracking uses a device installed in a vehicle, which transmits its location (coordinates) via GPRS (cell signal). A system operator can track that vehicle (or vehicles, for example from a fleet of buses), visualizing it on a map on screen.

The system must be able to manage the information from several vehicles at the same time, thus collecting it in a database, usually located on a server.

2.7.2. GPS Technical specifications

There are two different types of GPS tracking devices: active tracking devices and passive tracking devices. The active tracking devices have the ability to communicate through cellular or satellite networks to report their location in real time. Passive devices store GPS location, speed, heading and sometimes a trigger event. Once the vehicle returns to a predetermined point, the data is downloaded to a computer.

The devices that acquired the data available for this work, used active and passive tracking abilities: when a cellular network is available it transmits data to a server; when a network is not available the device stores data in internal memory and transmits stored data to the server later when the network becomes available again (Bartlett, 2009).

GPS devices use WGS-84 geographical coordinates system.

The margin of error of commercial vehicle GPS devices is about 15 meters in an open field. However, in an urban setting, the determination of a vehicle's position can be off by more than 50 meters, due to the signals bouncing off of obstacles like buildings, trees, or narrow streets, for example (Universidad Carlos III de Madrid, 2013).

3. Methods

The process of acquiring information from GPS dataset involves the design of logic algorithms, the design of the tool according to these algorithms and the programming itself.

This chapter shows the details regarding these phases.

3.1. Initial resources

3.1.1. OSM Plugin

OSM Plugin has been assembled as a Python Package, which means a "collection of modules" (Python Software Foundation, 2013), holding and maintaining the plugin data. The name of this package and the folder containing all programming is "osm". Inside the "osm" folder, several python modules and two additional folders containing specific procedures and styles are found.

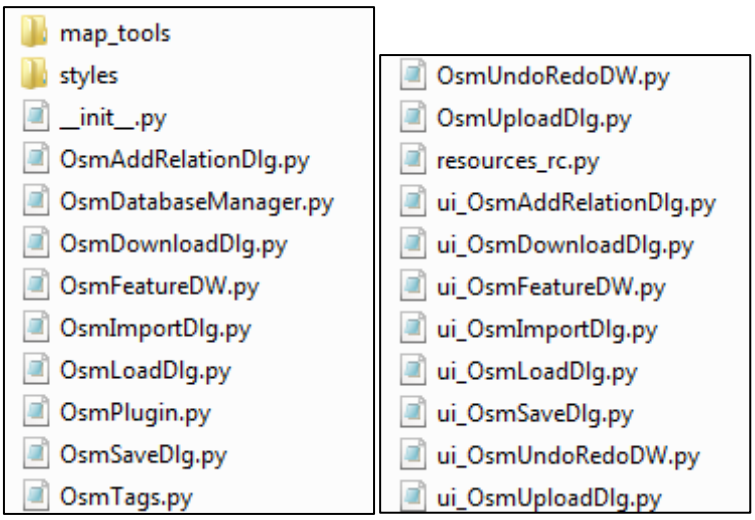


Figure 2.- List of files inside the OSM plugin folder

When QGIS starts, so do QGIS enabled plugins, like the OSM Plugin. The __init__.py module is then called automatically, which invokes the OsmPlugin.py module through its class “OsmPlugin”. This is the main class holding all the buttons, forms and procedures, and constructs the plugin itself, as a python object, creating the “python actions” that are presented to the user as buttons in the OSM toolbar.

Python OSM Plugin Graphical User Interface (GUI) is composed of six buttons, each of them pointing to specific Python modules. When a button is clicked, an action is invoked, calling another class in the corresponding module, which in turn calls a module with the GUI of the specific tool.

The button actions to be used in this GUI are defined in the main tool module. All GUI objects used, like buttons, combo styles, and the form itself, come from the PyQt4 modules, imported to the specific module at the beginning of the script.

As an example, the process from QGIS starting to the GUI appearing is shown for the “Load OSM from file” button.

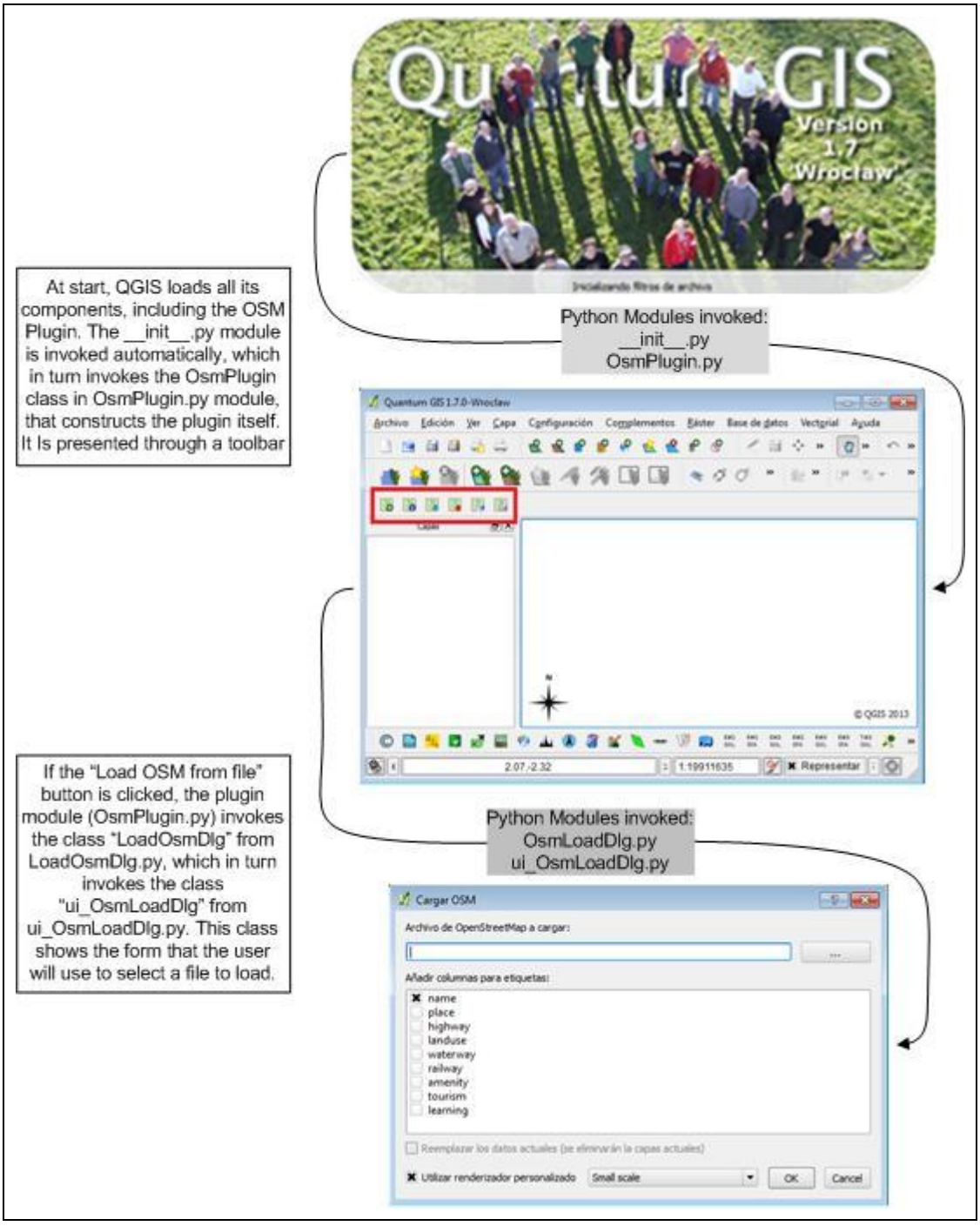


Figure 3.- Modules invoked when "Load OSM from file" button is clicked

3.1.2. Original data file

Vehicle GPS devices data to be used comes from SGAINNOVAR S.A. databases, from one of its customers in Portoviejo, Ecuador. This data is received

and stored in a PostgreSQL database that cannot be reached from external applications. So an excerpt was provided, that contains all important data. This excerpt was generated as a CSV (Comma Separated Values) file.

Data is sent by every vehicle GPS device, every minute. When an event like speeding is reported, data is sent too. So a new record is added to the database for each vehicle every minute, and when an event occurs.

A sample of this data is shown below, with 4 records.

```
560030;2012-10-01 06:00:57;0;-80.47162550;-1.05872900;.00;.00;Portoviejo;Washington y Calle Chone;12.92;2012-10-01 06:01:03;602
560030;2012-10-01 06:01:57;0;-80.47172350;-1.05871583;.00;.00;Portoviejo;Washington y Calle Chone;11.01;2012-10-01 06:02:04;602
560030;2012-10-01 06:02:11;0;-80.47186533;-1.05872300;.00;263.00;Portoviejo;Washington y Calle Chone;15.81;2012-10-01 06:02:17;61
560030;2012-10-01 06:02:57;0;-80.47180967;-1.05872617;.00;.00;Portoviejo;Washington y Calle Chone;6.21;2012-10-01 06:03:02;602
```

Attributes are separated by semicolons, and are explained in the next table.

Data in CSV file	**Description**
560030	Device ID
2012-10-01 06:02:11	Date and time
0	0 means data was real-time transmitted
-80.47186533	Longitude
-1.05872300	Latitude
.00	Speed, not always available
263.00	Direction or sens
Portoviejo	City, according to Longitude and Latitude
Washington y Calle Chone	Current adress, according to Longitude and Latitude
15.81	Distance from last recorded point
2012-10-01 06:02:17;61	Date and time record was stored in the database

Table 2.- Information provided by CSV original file

Attributes Device ID, Date and time, Longitude, Latitude are the only ones to be considered for the development of this work, as the objectives of this work intends to use only basic resources.

3.2. Results obtainment

Next sub-chapters show how the acquisition of average speed, maximum speed and road directions was oriented. This chapter includes solutions to logic issues that were found during the testing phase.

3.2.1. Maximum speed

Maximum speed in Ecuador is determined by the corresponding legal document: "*Reglamento General para la Aplicación de la Ley Orgánica de Transporte Terrestre, Tránsito Y Seguridad Vial*" (Comisión de Tránsito del Ecuador, 2011).

Article 192 mentions that within urban boundaries, maximum speed is 50 km/h for light vehicles, and 40 km/h for public transportation. As this tool is intended to be eventually used by common users, light vehicle limit was used, that is 50 km/h.

3.2.2. Average speed

Average speed is calculated for a given OSM way. For this, pairs of consecutive points that correspond to a specific way were used. Each point included information about its location and corresponding time, so speed can be calculated. The figure below shows some points corresponding to the red road, and for every pair of points, a speed value is calculated.

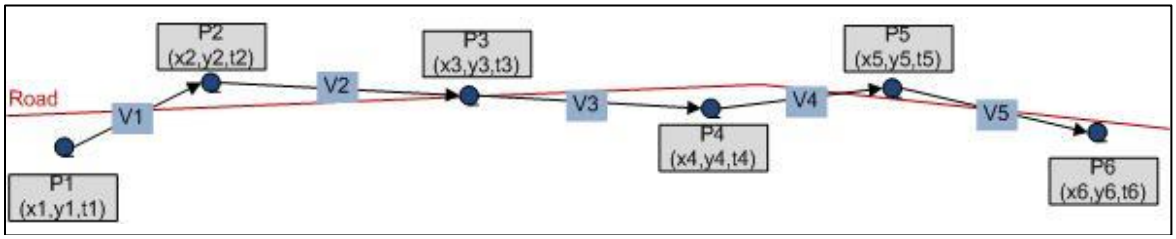


Figure 4.- Elements used to determine average speed

Average speed (AS) is calculated from location and time information, as follows:

$$AS = \frac{\sum_1^n \left(\frac{\sqrt[2]{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{t_{i+1} - t_i} \right)}{n}$$

Ecuador uses speed values measured in unit kilometers per hour (km/h), so location and time values have to be transformed to this unit system.

3.2.3. Road direction

Basic algorithm

As mentioned earlier, OSM ways may have the “oneway” tag to store information about road directions. If “oneway” tag value is “yes”, then it is a one way direction road from starting node to ending node; if value is “-1”, the direction is from ending node to starting node, and “no” means it is a two-way street. Roads that have no “oneway” tag are treated as two-way roads. Routing systems assume there is no restriction and use them in both directions.

As for Speed Average calculations, pairs of consecutive points were used. Points must be consecutive in time, which means the second point is a minute later (or less) than the first one. The tool evaluates, for each point of a specific pair, the distance from beginning and ending nodes, and determines the direction.

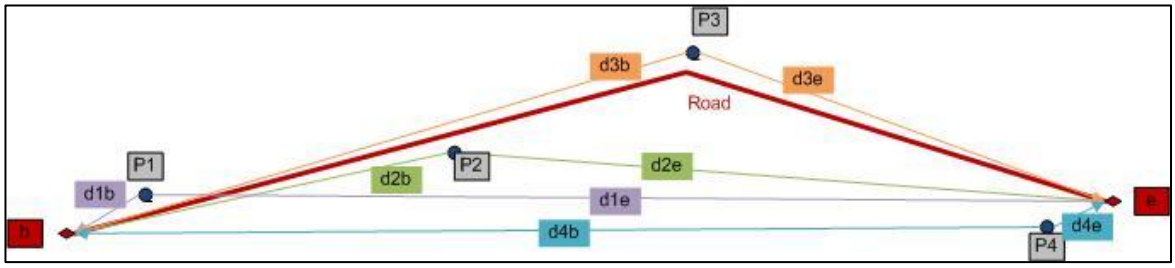


Figure 5.- Elements used to determine road directions

If the first point is closer to the beginning node than the second point, and the second point is closer to the ending node than the first one, then the road's direction is "from beginning node to ending node". For example, in the figure shown before, it is determined that P1 (Point 1) is closer to beginning node (b), and P2 is closer to ending node (e).

On the contrary, if the first point is closer to the ending node, and the second point is closer to the beginning node, then the road's direction is "from ending node to beginning node".

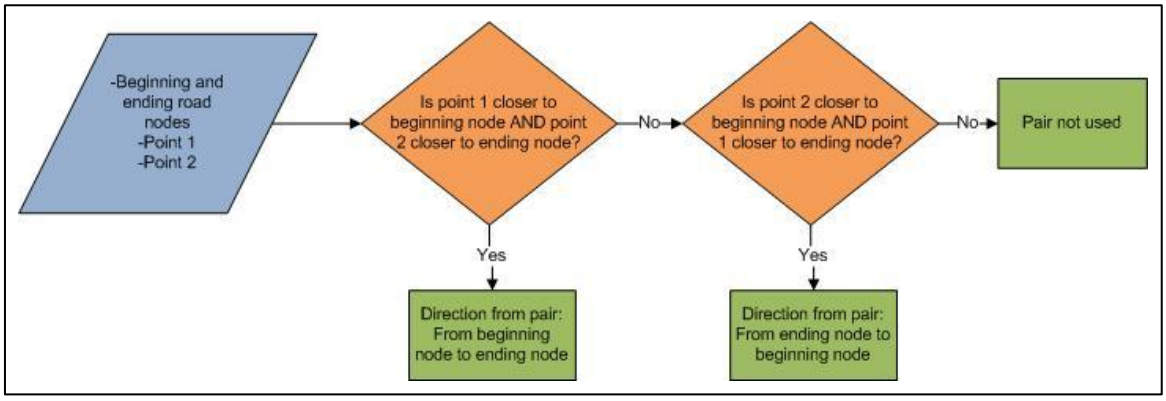


Figure 6.- Decision process for determining the direction of a road, from a pair of points

Roads are not always straight, and there is a minute of delay between point locations that could be used, for example, for a vehicle to go around a block, or making a U turn, or other situations that could confuse the tool. Because these cases might occur, all possible pairs are evaluated, and then percentages are given for each direction found. If a given direction obviously prevails, that direction is kept. If none of the directions prevail, then it is a two way road.

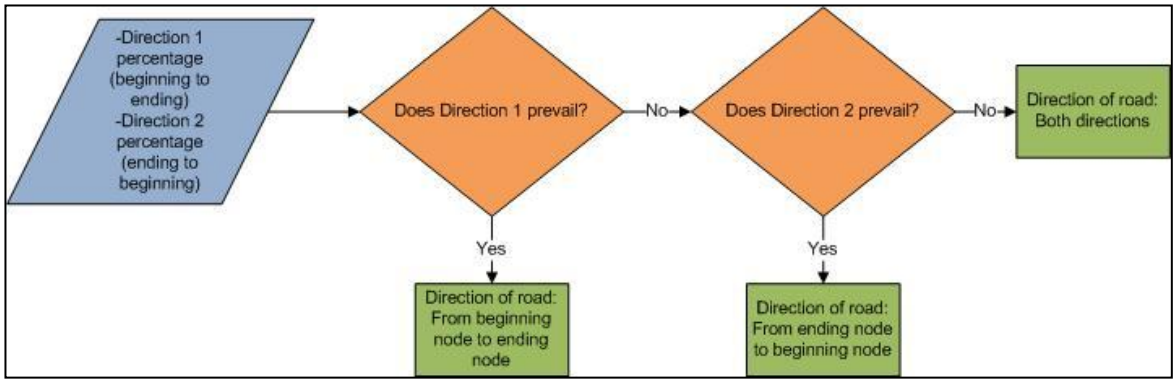


Figure 7.- Decision process for determining the direction of a road, from direction percentages

Algorithm considerations

The algorithm to find the directions of roads mentioned earlier assumes vehicles go over streets randomly, and there is enough data for each road to process. But the data provided for this work does not come from random vehicles; instead it comes from a set of urban buses that go over roads according to a predetermined schedule. So roads that are two ways could be used only in one direction, other roads are not used at all.

If the directions are determined as shown in the basic algorithm, it could happen that a road is only used in one direction by urban buses, so the tool assigns the value “yes” or “-1” to the “oneway” tag. If it is actually a two-way street, then that direction is wrong. A routing tool using “oneway” tag information would not allow cars to use the road in both directions.

Because of this, the basic decision algorithm must be modified, so actions taken do not provoke routing systems to throw results with errors.

All possibilities are shown in the next table, for a given road.

Tag	Scenario	Preferred action
"no" (two ways)	Buses go over road two ways	Do nothing, information is updated
	Buses go over road only one way	Do nothing, buses might not go over a two-way road both ways
"yes" or "-1" (one way)	Buses go over road two ways	Update, it is a two way road
	Buses go over road only one way, same way	Do nothing, information is updated
	Buses go over road only one way, opposite way	Update to two-way road (case explained below)
no "oneway" tag	Buses go over road two ways	Update, it is a two-way road
	Buses go over road only one way	Do nothing, buses might not go over a two-way road both ways

Table 3.- Actions regarding the modified algorithm

If a road is one way (“yes” or “-1” tags), and the tool finds only the opposite direction, it could imply that direction has been changed to that road, from one way to the opposite way. But it could also imply that urban buses go over it the opposite way, being a two-way street. In this case the tool should assign the “no” tag, because of the cases explained before, the second one does not restrict vehicles to use that road.

In all tag cases, if there are no vehicles going over a given street, no action is taken.

Need for external assistance

As mentioned, the algorithm can determine a probable road direction, but not solve all logic cases, due to data limitations. The cases explained before propose logic solutions to the cases that can be found, but not all of them can be solved this way, because doing so would provoke some updates to be incorrect.

Therefore, it is required that a person with road direction knowledge of a given city provides external assistance to the tool. The proposed way to achieve this is including an approval stage to updating process.

3.3. Tool design

The tool to be developed has to deliver an OSM file processed according to the GPS data and the user criteria. Therefore GUI must guide the user from opening an OSM file and a GPS data file, until processing the whole file.

3.3.1. Tool design

To create the new tool, original OSM plugin was modified, so users could still use buttons and procedures as they used to. Original tools remained the same, and a new button was created. From this button, forms appear to provide basic configuration settings. After these settings, processing and solving start.

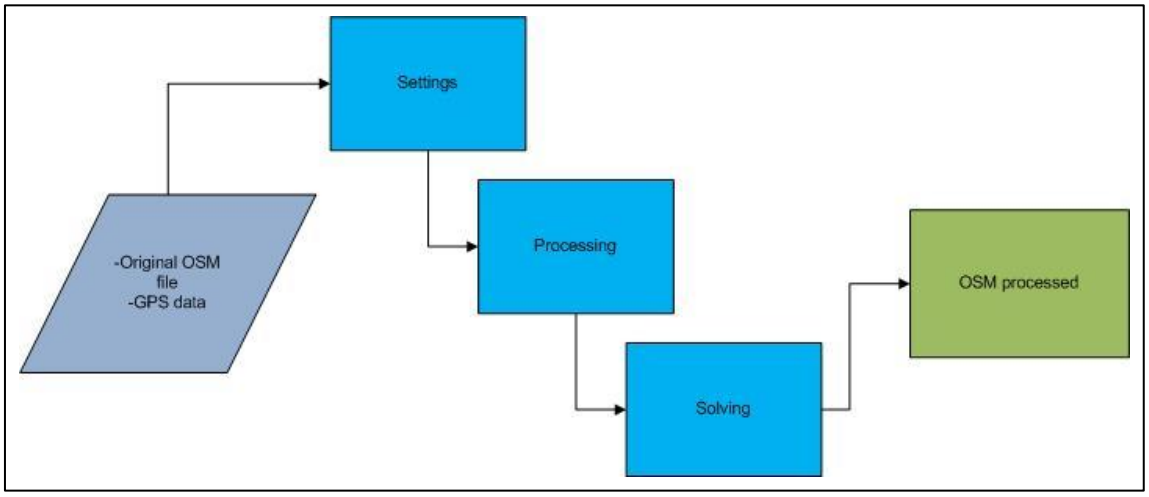


Figure 8.- Basic tool work flow

Settings refer to selecting, opening and displaying an OSM file, and opening a CSV file containing GPS data. Besides this, users might decide to include the maximum speed tag for considered roads. After this, the tool processes the files and determines a list of proposed road directions, and average speeds, generating a set of possible changes that users can then solve.

These three stages are not common to all the information to be determined, so they are explained in the next figures.

Maximum speed

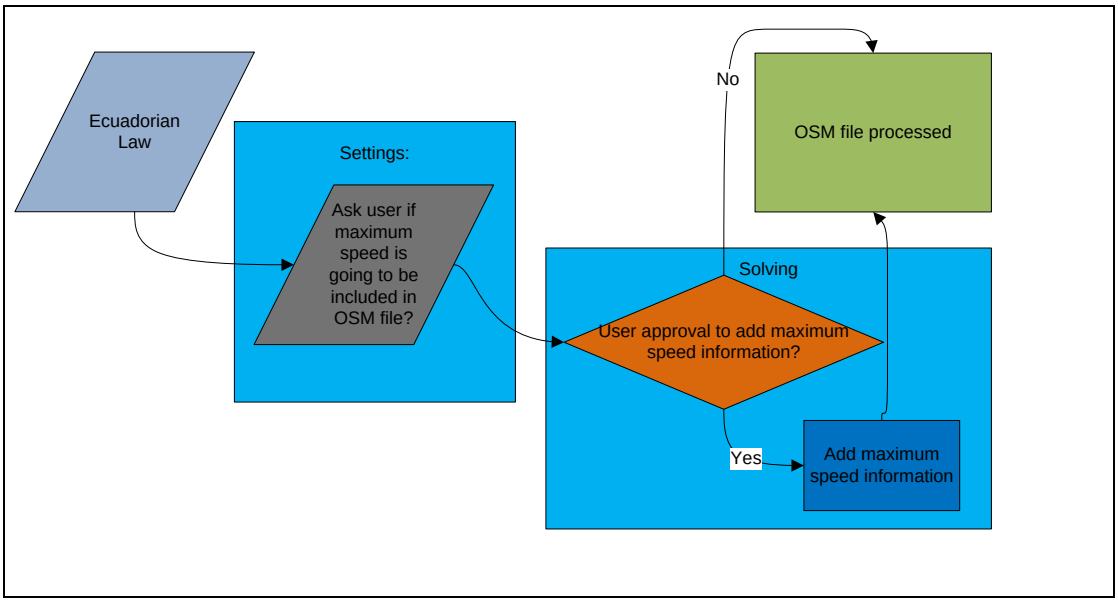


Figure 9.- Maximum speed information flow

If the maximum speed information is going to be used, it should be used for all city streets to be processed. User should decide this before solving. No processing is needed, only setting and solving.

Average speeds and road directions

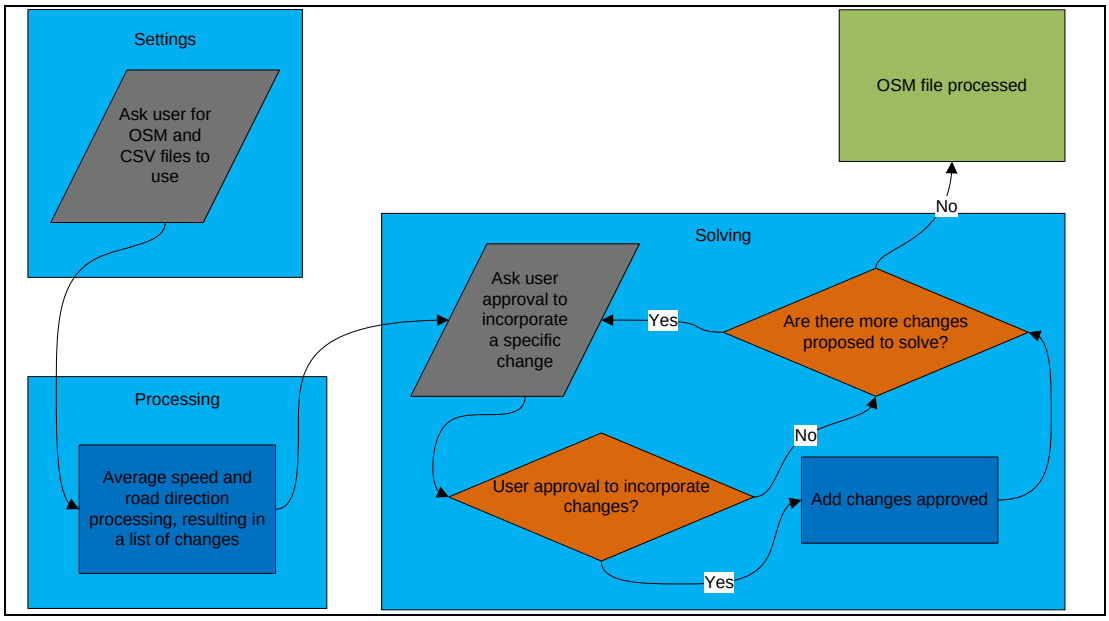


Figure 10.- Average speed and road direction information flow

Tool GUI

A form is generated every time user interaction is needed. As shown in the figures about information flows, users should be asked for interaction in settings and in solving stages, so two forms are generated.

The settings form asks the following:

- OSM file to use
- CSV file to use
- Approval to add maximum speed to all urban roads considered

The solving form asks for the following approvals:

- Include the proposed average speed
- Change to proposed road direction

This last form must be shown to users simultaneously with map information. Users should be able to visualize the road being considered, and the changes being proposed, to approve (or not) those changes.

3.3.2. Python programming

As said before, all original buttons and their corresponding procedures remained the same. All additional programming was made creating new scripts or modifying the original scripts, only adding required programming statements or functions.

Python scripts

OSM package consists of various python scripts necessary to develop all procedures. At starting, __init__.py is invoked. This script was modified to provide

users information regarding the modified status of the plugin. New name to be shown to users is “OpenStreetMap plugin, Modified for GPS data processing”. Also new description is "Viewer and editor for OpenStreetMap data (Modified UNIGIS)".

OsmPlugin.py is invoked to construct the plugin, as a python class. Within this python script a new button was created, that launches new procedures to process GPS data.

New toolbar presents a new button, as shown in the next figure.

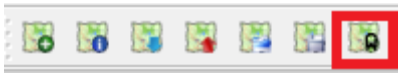


Figure 11.- New toolbar, including OSM process button

A new icon was created for this button, as a “jpg” file.

Figure 12.- New button

“__init__” and “OsmPlugin” were the only original scripts modified; only adding references to new scripts. The rest of the programming was included in five new scripts that were created, and organized in two categories: processing scripts and solving scripts. Main processing and solving classes are invoked from “OsmPlugin” through the new button. As shown in the code below, first python class invoked is GPSProcess, which results in all proposed changes to approve later, and the second class is GPSSolve, that goes through those changes and asks the user for approval, one by one.

```
def processGPSData(self):
    """Main procedures to process GPS data and an OSM file
    UNIGIS
    """

    self.processedData=GPSProcess(self)
    # continue only if OK button was clicked
```

```
if self.processedData.exec_()==0:
    return

for i in self.processed:
    self.solvedData=GPSSolve(self, i)
    # continue only if OK button was clicked
    if self.solvedData.exec_()==0:
        return
```

The rest of the scripts created are called from these two classes, and are explained in the next table.

Category	Name	Class	Description
Processing	UNIGIS_ ProcessMain	GPSProcess	This class calls the form to provide initial information. It uses original OSM OsmLoadDlg.py to load OSM files, and a modified version of this script to open CSV files. Once opened, it also invokes the processing class, updateDirectionsAndSpeed.
	UNIGIS_ ProcessTools	updateDirections AndSpeed	All functions to process average speed and road direction are contained by this class. It uses OSM and CSV files determined by the GPSProcess class, resulting in a list of changes to be approved later.
	UNIGIS_ ProcessUI	Ui_processGPS	Creates the form to ask initial settings: OSM and CSV files, and if maximum speed limit is included in processed OSM file.
Solving	UNIGIS_ SolveMain	GPSSolve	Takes the list of proposed changes from GPSProcess class, and guides the user to go over it.
	UNIGIS_ SolveUI	Ui_solveGPS	Creates the form to ask for approval: This form also sets the map canvas to show the features involved, and indicates actual and proposed road directions.

Table 4.- Scripts created and Python Classes involved

This classes and scripts flow is presented in the next figure.

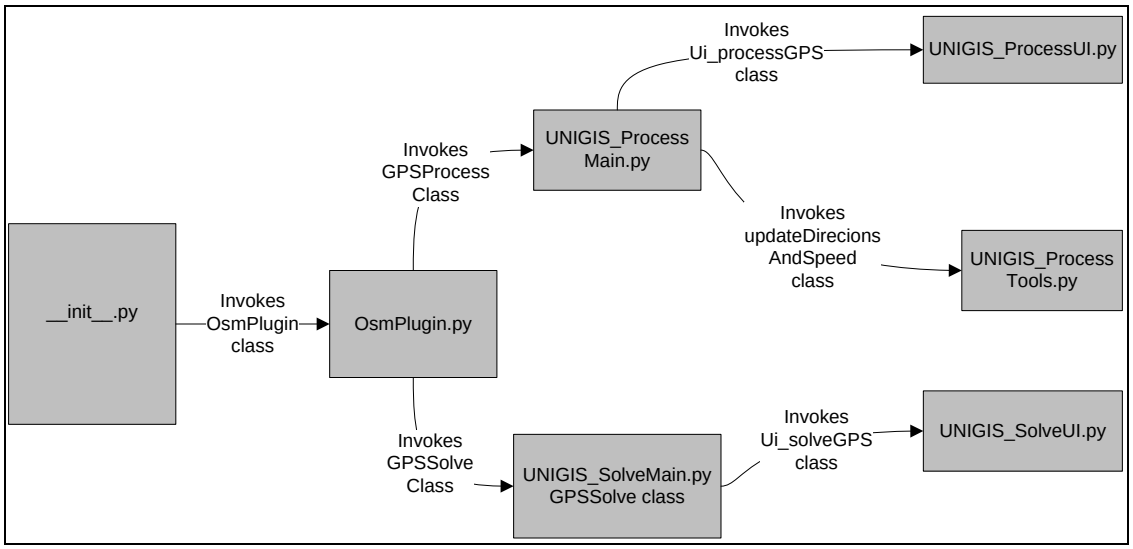


Figure 13.-Scripts used and classes invoked

This classes flow share information between scripts. The next figure shows this information flow.

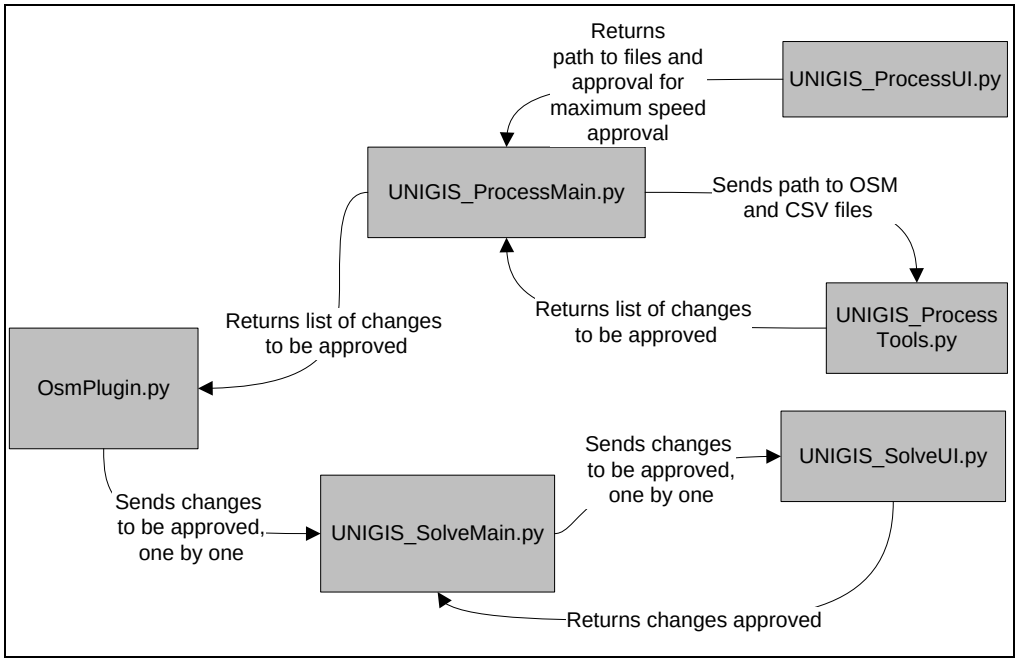


Figure 14.- Information flow between python scripts

Python resources used

The tool generated uses modules from the standard library for Python 5.1 version, or from installed modules provided by QGIS. Each of the modules used is imported into a specific script where it is used. Those scripts are listed in the next table.

Module name	Imported by	Module description
CSV	UNIGIS_ProcessMain	Used to parse CSV files. Handles CSV reading and writing capabilities.
xml.etree. ElementTree	UNIGIS_ProcessMain UNIGIS_SolveMain	Creates a "Tree" readable by python from a XML file, can be used for an OSM file too. Handles reading and writing capabilities.
math	UNIGIS_ProcessTools UNIGIS_SolveUI	Provides mathematical functions to scripts.
qgis.gui	UNIGIS_ProcessTools UNIGIS_SolveUI	This class allows canvas interaction, like zooming or creating temporary objects.
PyQt4 (QtCore and QtGui)	All scripts.	Qt is used for GUI implementation, and for message used in testing phase.

Table 5.- Modules used in generated scripts

3.4. Testing phase

The tool to be developed in this work was compulsory oriented to the resources available, especially GPS data provided by a specific company. This issue reduces the possibilities to test the tool, as only GPS vehicle data for a buses fleet was provided.

Therefore, the tool was considered ready as it produced accurate results with available data. For this matter the tests described next were conducted.

#	Test	Description
1	Simplified test	A scenario with unreal coordinates values was created, so values can be tested in a simplified way. This test allows algorithm issues to be solved.
2	Simplified georeferenced test	This scenario shows possible problems when dealing with simplified real coordinates (the tool works with GPS dataset coordinates, latitude and longitude).
3	Real simplified tests	Samples of few streets were tested against a sample of real GPS dataset. Algorithm can be fully tested.
4	Real test	Complete tests were conducted, using full Portoviejo OSM streets and samples of GPS data. These tests show performance issues.

Table 6.- Tests description

Besides algorithm issues, programming issues were also corrected as they were detected.

3.4.1. Simplified test

Streets and GPS coordinates are simple and unreal, as shown next. They are contained in an OSM and a CSV file. Length unit is meters, so a temporary change of programming was made to perform speed calculation from these coordinates.

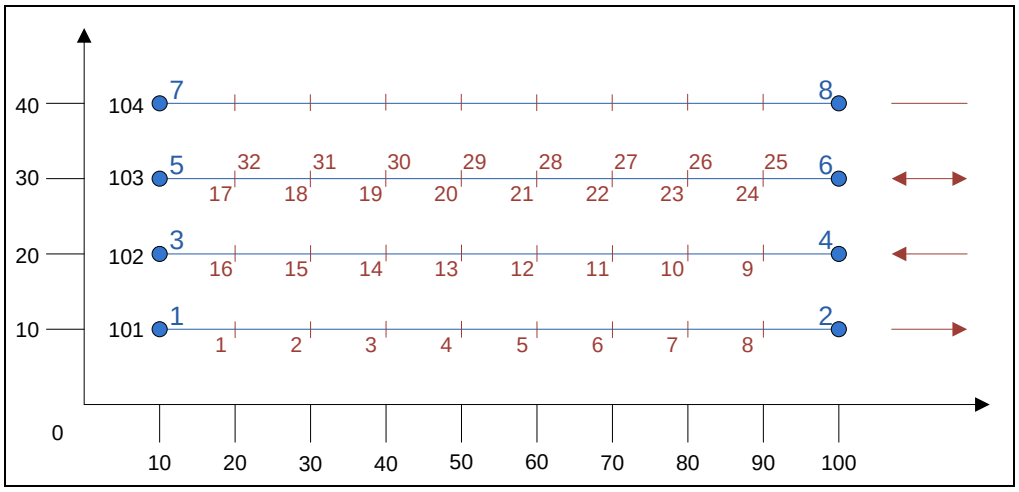


Figure 15.- Simplified Test data

Each GPS point (shown in red from 1 to 32) is given a different time, with a minute of delay. The last road does not have any corresponding points, as a

control road that must not be changed. Expected results are directions as shown in red arrows. Average speed is ${}^{10\ m}/_{1\ min} = 600\,\frac{m}{h} = 0.6\,\frac{km}{h}$.

3.4.2. Simplified georeferenced test

This test is very similar to the Simplified test, but GPS data was transformed to values that could be found in real life (latitude and longitude within Portoviejo).

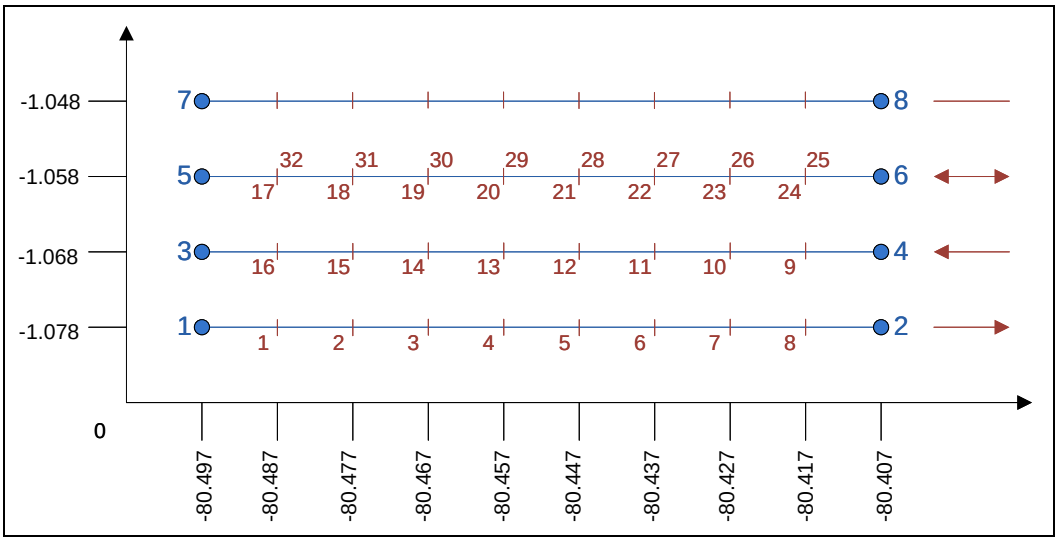


Figure 16.- Simplified Georeferenced test data

GPS points in CSV file are distributed as GPS points for Simplified test. Expected results are directions as shown in red arrows in Figure 17. Exact average speed depends on latitude, but is close to ${}^{0.01°}/_{1\ min} = 60.72\,\frac{km}{h}$.

3.4.3. Real simplified test

This test was conducted for five different small sets of real streets of Portoviejo, from OpenStreetMap. These streets were tested against a set of GPS vehicle data from only two months, so CSV file size does not provoke performance problems.

Average speed and directions depend on each street and available data.

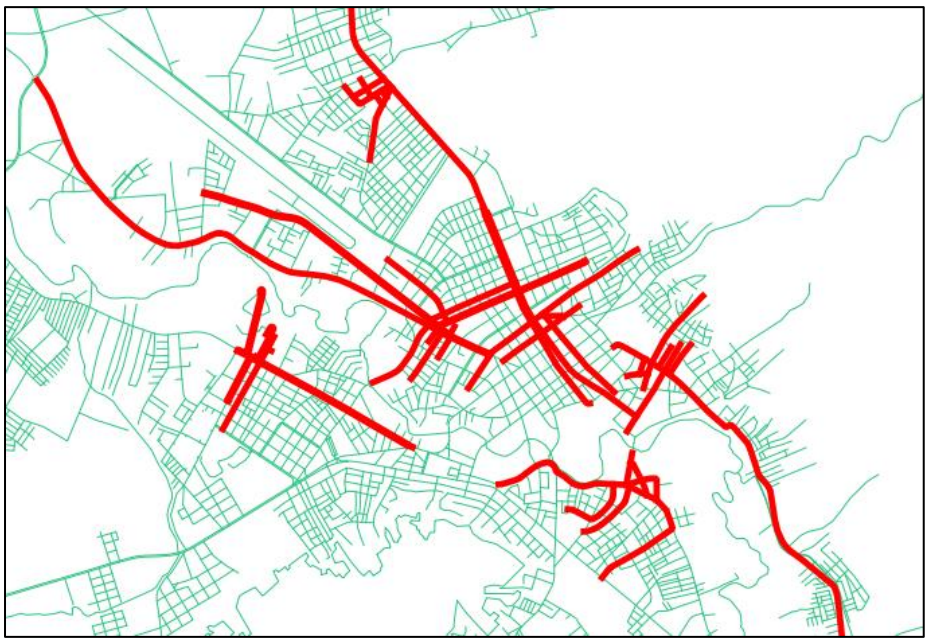


Figure 17.- Streets considered for Simplified real test

3.4.4. Real test

All algorithm and procedure should have been solved to this point. So performance tests were performed to prove capacity of the tool under normal conditions of hardware.

3.4.5. Checklist

For each test, a checklist was used to control results. Four scenarios were considered, regarding the “oneway” tag, and are shown in Appendix 1.

4. Results

This chapter shows the results of the testing phase.

4.1. Testing phase

4.1.1. Simplified test

This test is intended to review general flow of the tool.

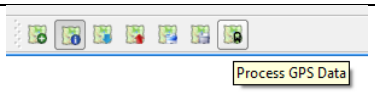
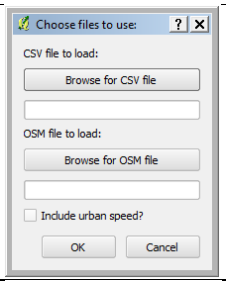
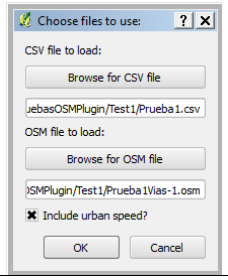
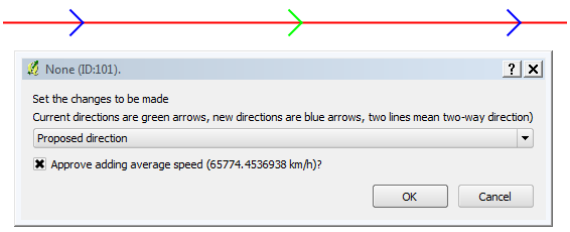
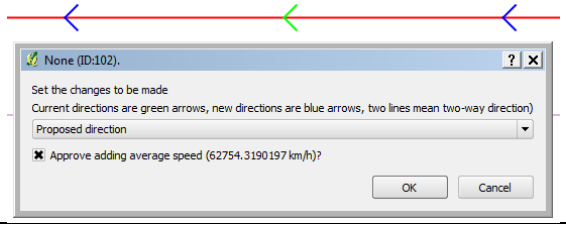
Stage	Event	Event screen
1	Click new button	
2	Settings form appears	
3	Choose CSV and OSM files. Check inclusion of urban speed limit. Click “Ok” to start and processing begins.	
4	First change to approve appears on screen. Road involved is highlighted. Also, road name and ID are shown as form title (“None” if there is no name). Average speed and road directions are shown as expected.	
5	Check for inclusion of approved changes. Click “Ok” button. Repeat until all changes are reviewed.	

Table 7.- Events of the first scenario for Simplified test

This flow remains the same for all tests.

OSM file changes are shown below, using XML notepad tool “Compare XML files”. Text highlighted in yellow shows xml code added, and green shows changes, comparing “way” identified by the “id” tag.

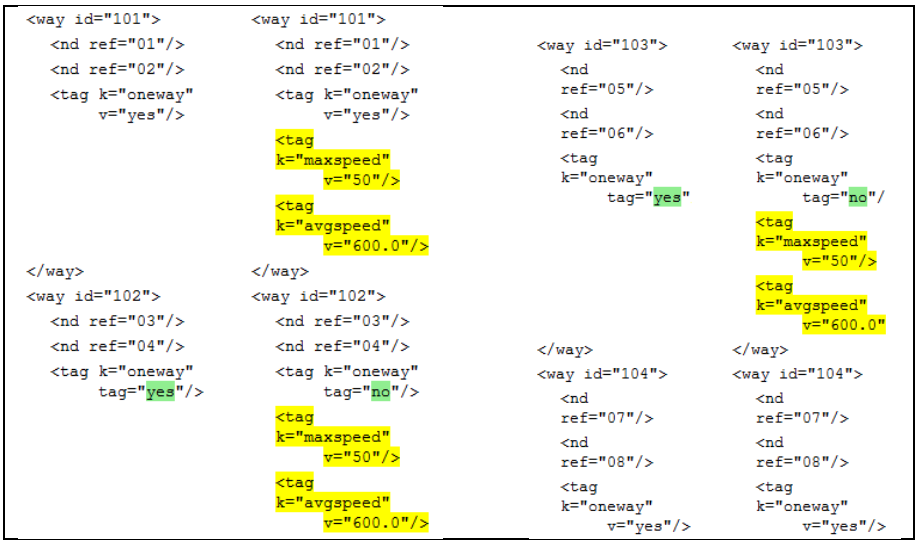


Figure 18.- Results of first test, “yes” value for “oneway” tag.

Changes are made as expected.

This test resulted in programming changes and algorithm changes, which were made until the checklist used was completely positive.

4.1.2. Simplified georeferenced test

This test is similar to the first one, but coordinates are now intended to be real, keeping the simple way distribution of the first test.

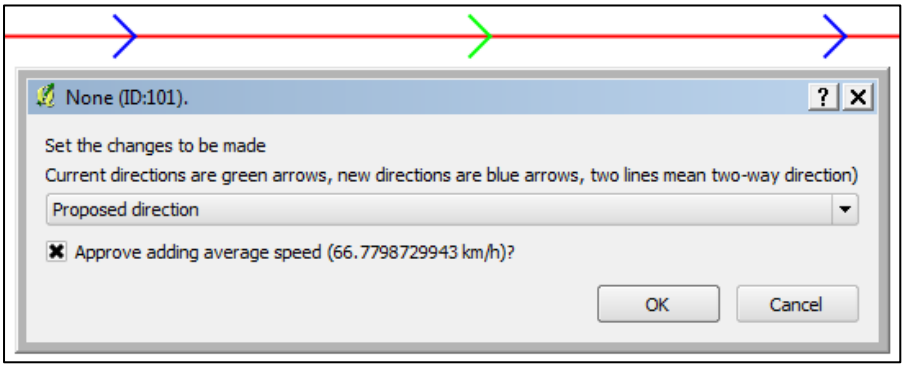


Figure 19.- Example of a change for Simplified georeferenced test

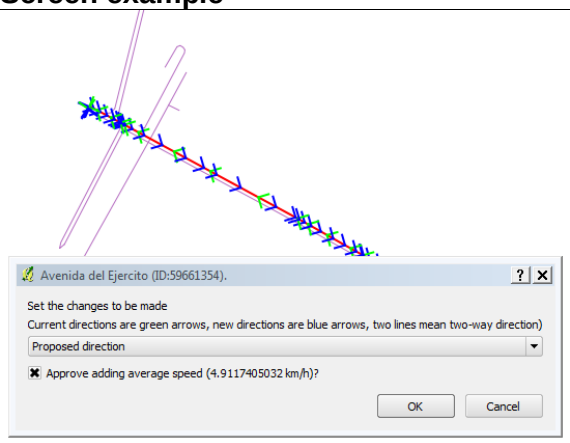
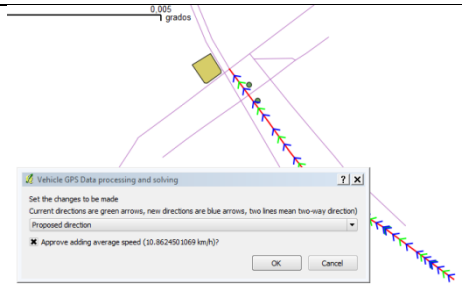
As for the first test, changes in OSM file are shown next.

Figure 20.- Results of second test, “yes” value for “oneway” tag.

Direction changes and average speed calculations are changed as expected.

4.1.3. Real simplified test

Five random excerpts of roads were tested.

Excerpt	Screen example	Example of OSM file changes
1		<tag k="name" v="Avenida del EjÃ©rcito"/> <tag k="oneway" tag="yes"/> <tag k="source" v="Yahoo Aerial Imagery"/> </way> <tag k="name" v="Avenida del EjÃ©rcito"/> <tag k="oneway" tag="no"/> <tag k="source" v="Yahoo Aerial Imagery"/> <tag k="maxspeed" v="50"/> <tag k="avgspeed" v="4.9117405032"/> </way>
2		<nd ref="493911468"/> <nd ref="493916620"/> <nd ref="491572937"/> <tag k="highway" v="residential"/> <tag k="name" v="9 de Octubre"/> <tag k="source" v="Yahoo Aerial Imagery"/> </way> <nd ref="493911468"/> <nd ref="493916620"/> <nd ref="491572937"/> <tag k="highway" v="residential"/> <tag k="name" v="9 de Octubre"/> <tag k="source" v="Yahoo Aerial Imagery"/> <tag k="maxspeed" v="50"/> <tag k="avgspeed" v="2.45866555955"/> <tag k="oneway" v="-1"/> </way>

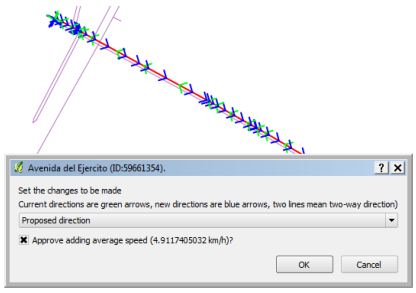
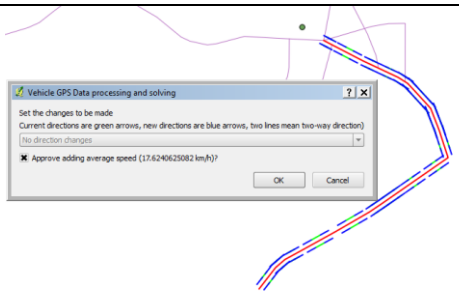
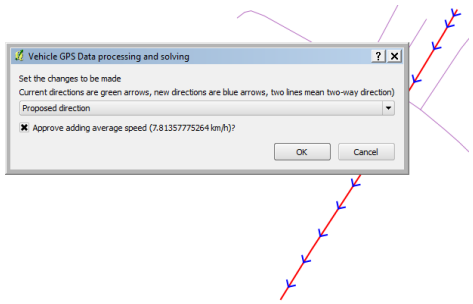
Excerpt	Screen example	Example of OSM file changes
3		<nd ref="475360527"/> <tag k="highway" v="secondary"/> <tag k="name" v="Avenida ManabÃ"/> <tag k="oneway" tag="yes"/> <tag k="source" v="Yahoo Aerial Imagery"/> 'way> <nd ref="475360527"/> <tag k="highway" v="secondary"/> <tag k="name" v="Avenida ManabÃ"/> <tag k="oneway" tag="-1"/> <tag k="source" v="Yahoo Aerial Imagery"/> <tag k="maxspeed" v="50"/> <tag k="avgspeed" v="0.4740855869</way>
4		<nd ref="479749493"/> <nd ref="479749282"/> <nd ref="479749284"/> <tag k="highway" v="tertiary"/> <tag k="name" v="Las Acacias"/> <tag k="oneway" v="no"/> <tag k="source" v="Yahoo Aerial Imagery"/> </way> </osm> <nd ref="479749493"/> <nd ref="479749282"/> <nd ref="479749284"/> <tag k="highway" v="tertiary"/> <tag k="name" v="Las Acacias"/> <tag k="oneway" v="no"/> <tag k="source" v="Yahoo Aerial Imagery"/> <tag k="avgspeed" v="17.6240625082 </way> </osm>
5		ref="497257723"/> <tag k="highway" v="residential <tag k="name" v="Eloy Alfaro"/> <tag k="source" v="Yahoo Aerial Imagery"/> ref="497257723"/> <tag k="highway" v="residential <tag k="name" v="Eloy Alfaro"/> <tag k="source" v="Yahoo Aerial Imagery"/> <tag k="maxspeed" v="50"/> <tag k="avgspeed" v="7.813577752 <tag k="oneway" v="-1"/>

Table 8.- Results of third test.

Real simplified tests resulted in a series of compulsory corrections made to code, as many issues were detected concerning programming errors that did not emerge in the tests before.

4.1.4. Real test

For this test, some excerpts were created from CSV and OSM complete files, to measure the amount of data a desktop computer can handle.

The goal of the tool is to process all Portoviejo OSM dataset so only two files were created. The first one was a small excerpt, to be tested first, so possible procedural errors could be detected before having to wait the time needed to process the complete file. The second one was the entire file downloaded.

File type	File name	File size (KB)
Osm	Portoviejo0010k	12
Osm	Portoviejo3300k	3299
Csv	CSV000030	28
Csv	CSV000200	190
Csv	CSV001000	996
Csv	CSV010000	10916
Csv	CSV100000	102210
Csv	CSV380000	382729

Table 9.- Files used for Real test

All CSV were tested against OSM files. Programming errors were still found, during this phase, and were solved.

Full CSV file containing GPS data is a large file, and not all of its content is usable. From this file all data concerning night hours was removed, that is from 23 p.m. to 6 a.m. (outside working hours), and only was kept data for a period of time of two months. From this new file, smaller files were created: (approx.) 30 KB, 200 KB, 1000 KB and 10000 KB, that were tested against a full Portoviejo OSM dataset.

As CSV files increase, processing time increased as well. These processing times are shown next, when processed against complete Portoviejo OSM dataset.

CSV File	Processing time
CSV000030	3 minutes
CSV000200	16 minutes
CSV001000	58 minutes
CSV010000	3 hours
CSV100000	Over 10 hours
CSV380000	Memory error

Table 10.- Processing time for Real Test

Large CSV files are difficult for normal computers to handle, when processed with the tool created.

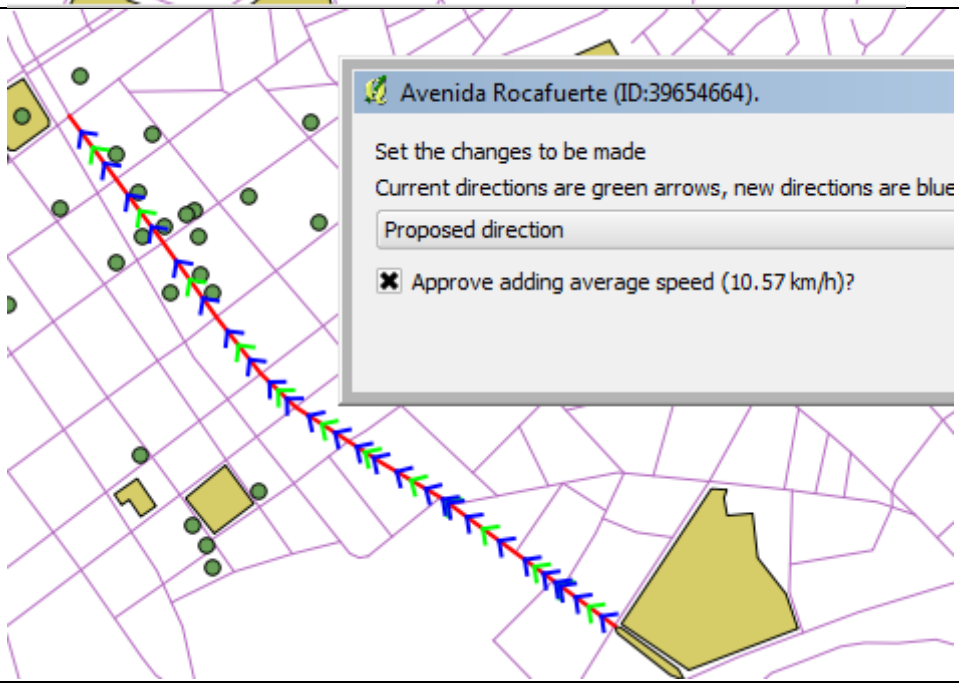
4.2. Testing phase synthesis

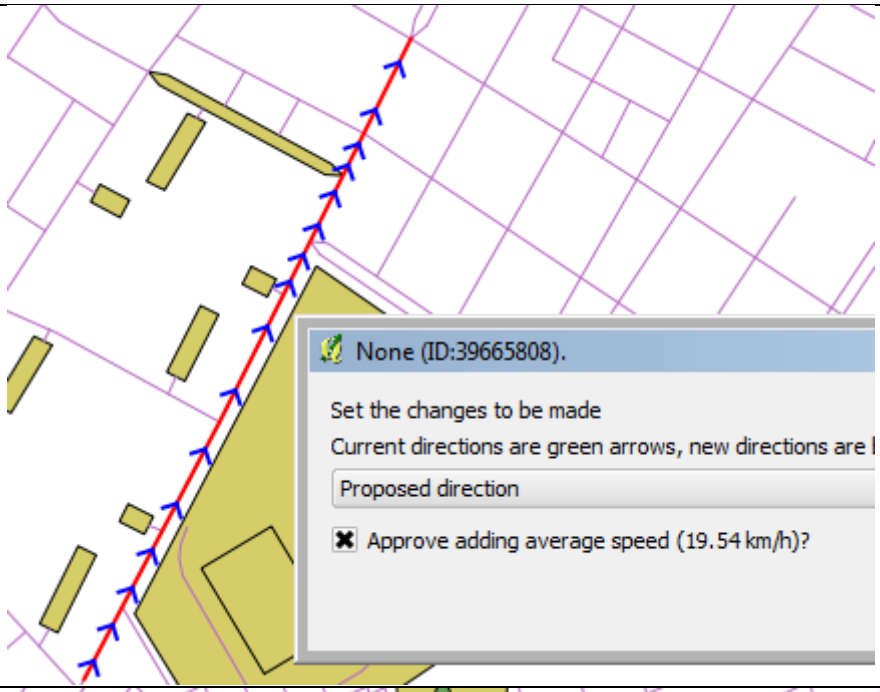
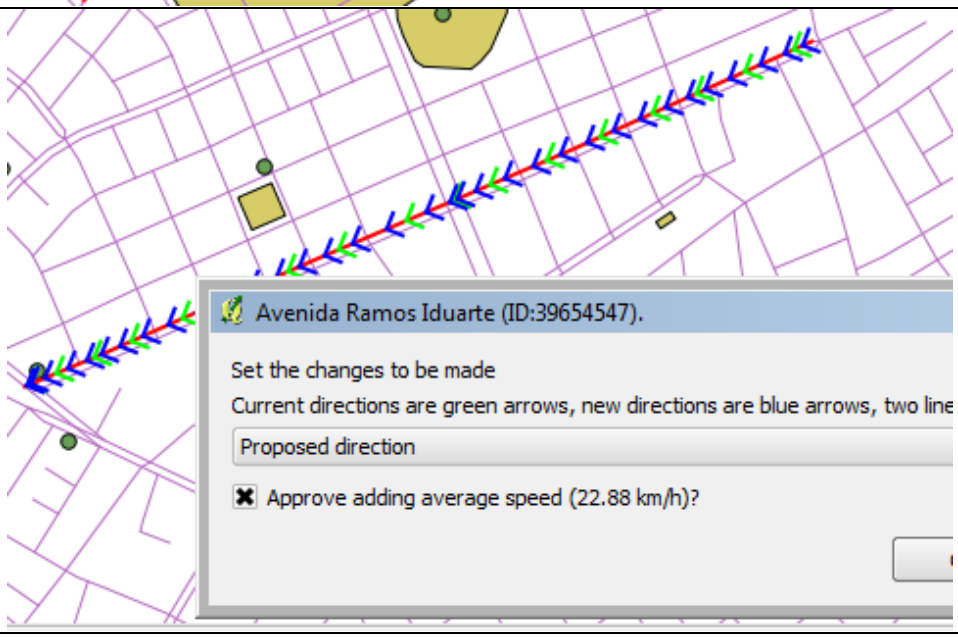
Testing phase permitted to expose how the tool worked with real data, and how its performance achieved the results expected, regarding average speed and road direction calculations. A brief synthesis is presented in this chapter.

The cases intend to present logical results, which mean that are presented events where the tool works in a coherent way, as expected. All other cases are analyzed in the Discussion chapter.

4.2.1. Average speed

From tests conducted with real Portoviejo road OSM data, example average speeds are shown in the table below.

	Print screens
1	
2	

3	 None (ID:39665808). Set the changes to be made Current directions are green arrows, new directions are Proposed direction Approve adding average speed (19.54 km/h)?
4	 Avenida Ramos Iduarte (ID:39654547). Set the changes to be made Current directions are green arrows, new directions are blue arrows, two line Proposed direction Approve adding average speed (22.88 km/h)?

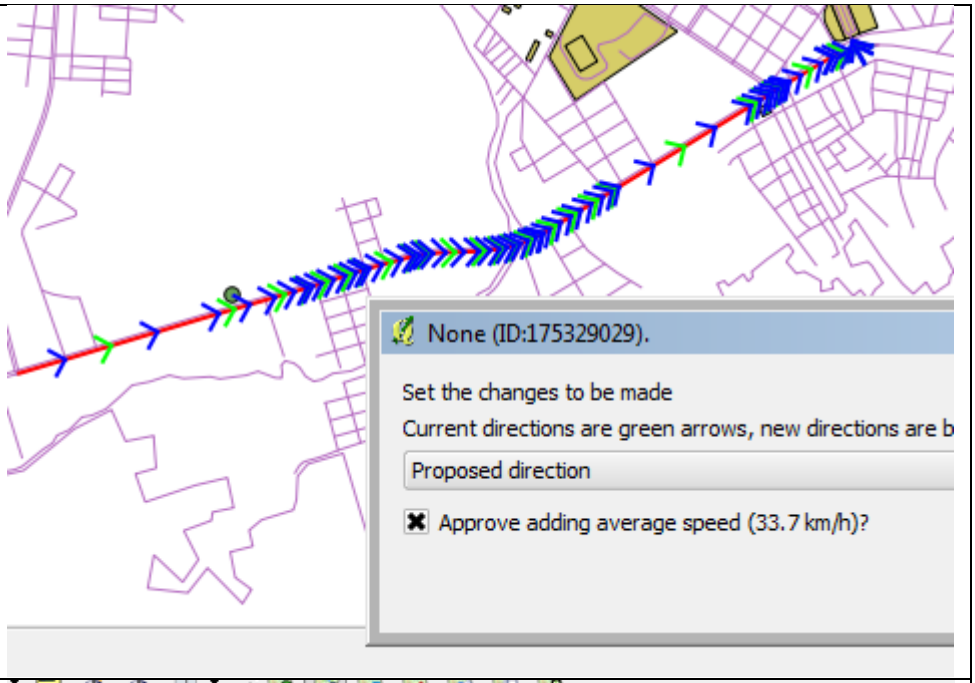
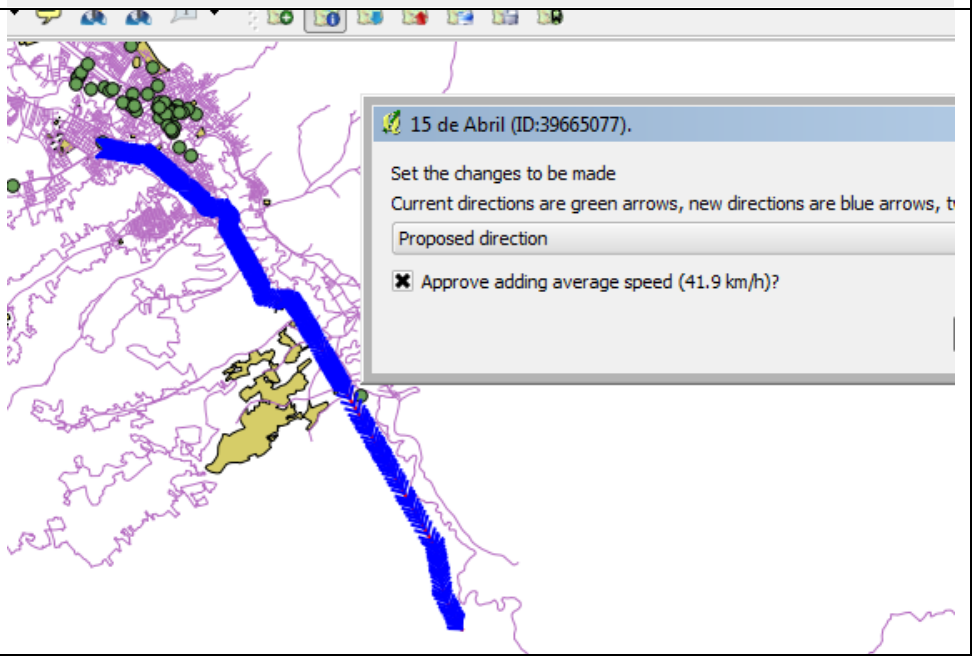
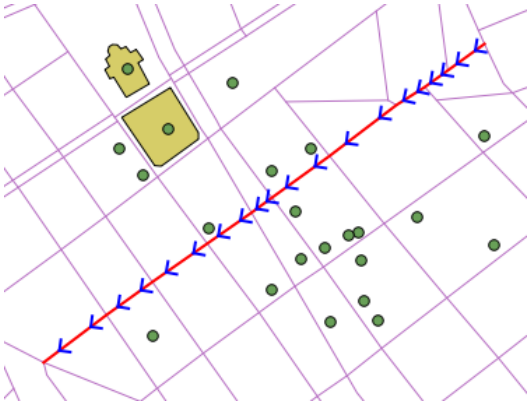
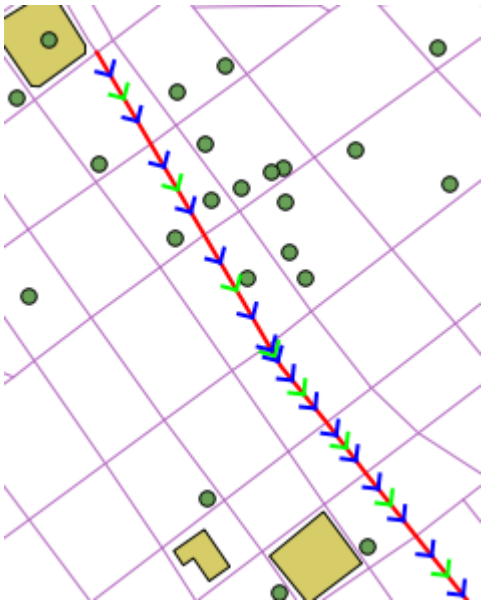
5	
6	

Table 11.- Calculated average speed examples

These examples have been ordered increasingly, according to speed average value. Low values (cases 1 to 4) are found downtown, where heavy traffic can be expected. Cases 5 and 6 correspond to larger roads so higher speeds values are found.

4.2.2. Road directions results

Algorithm for determine road directions functions according to logical flow from which the tool was designed and created. This implies that always colored arrows appear onscreen as expected, proposing a change to be approved by final user.

Road and changes proposed	OSM addition (yellow)
ID: 59640277 	<way id="59640277" user="jose_ecu" uid="244733" visible="true" version="1" changeset="4751652" timestamp="2010-05-19T20:30:49Z"> <nd ref="491525468"/> <nd ref="491525470"/> <nd ref="491525471"/> <nd ref="493911404"/> <nd ref="491572934"/> <nd ref="491572935"/> <nd ref="493911461"/> <nd ref="493911508"/> <nd ref="493911468"/> <nd ref="493916620"/> <nd ref="491572937"/> <tag k="highway" v="residential"/> <tag k="name" v="9 de Octubre"/> <tag k="source" v="Yahoo Aerial Imagery"/> <tag k="oneway" v="-1"/>
ID 59640275: 	<way id="59640275" user="jose_ecu" uid="244733" visible="true" version="2" changeset="4752869" timestamp="2010-05-19T22:43:17Z"> <nd ref="491572910"/> <nd ref="491572934"/> <nd ref="491572942"/> <nd ref="493911408"/> <nd ref="491572914"/> <nd ref="491572915"/> <nd ref="491572917"/> <nd ref="493911428"/> <nd ref="493911425"/> <nd ref="491572918"/> <nd ref="491572920"/> <nd ref="491572921"/> <tag k="highway" v="secondary"/> <tag k="name" v="Avenida Morales"/> <tag k="oneway" v="yes"/> <tag k="source" v="Yahoo Aerial Imagery"/> <tag k="avgspeed" v="5.26"/>

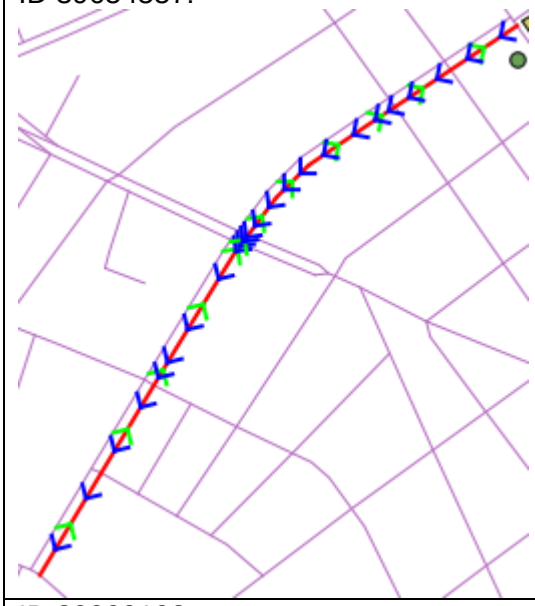
<table>
<tr>
<td>ID 39654557:
  </td>
<td><way id="39654557"
user="jose_ecu"
uid="244733"
visible="true"
version="9"
changeset="4751652"
timestamp="2010-05-19T20:31:02Z">
<nd ref="491489913"/>
<nd ref="491495114"/>
<nd ref="491489831"/>
<nd ref="475360538"/>
<nd ref="475360539"/>
<nd ref="475366341"/>
<nd ref="475366340"/>
<nd ref="475360540"/>
<nd ref="475360541"/>
<nd ref="475360542"/>
<nd ref="491525365"/>
<nd ref="491525436"/>
<nd ref="493911402"/>
<tag k="highway"
v="secondary"/>
<tag k="name"
v="Avenida Alhajuela"/>
<tag k="oneway"
v="-1"/></td>
</tr>
<tr>
<td>ID 39909192:
  </td>
<td></way>
<way id="39909192"
user="LLAQWA"
uid="77114"
visible="true"
version="4"
changeset="2557140"
timestamp="2009-09-21T13:27:23Z">
<nd ref="479153534"/>
<nd ref="479153536"/>
<nd ref="479153539"/>
<nd ref="479153544"/>
<nd ref="479153631"/>
<nd ref="479153632"/>
<nd ref="479161158"/>
<nd ref="479153633"/>
<nd ref="479153634"/>
<nd ref="479153635"/>
<nd ref="479153637"/>
<nd ref="504065392"/>
<nd ref="479153638"/>
<nd ref="504065429"/>
<nd ref="479153619"/>
<nd ref="479153639"/>
<nd ref="479153640"/>
<nd ref="479153641"/>
<nd ref="479153642"/>
<tag k="highway"
v="residential"/:
<tag k="source"
v="Yahoo Aerial Imagery"/>
<tag k="oneway"
v="no"/></td>
</tr>
</table>

Table 12.- Determined road direction examples

This examples show how the tool handles given conditions, and integrates, as commanded by the user, road directions into the OSM file. For the examples shown, always the proposed option was chosen.

5. Discussion

As mentioned earlier, not always the tool could achieve results as expected. This chapter exposes the reasons found for the tool not to perform correctly, or special conditions for a result to be inaccurate.

It is convenient to mention that there was no field work developed for this work (logistic issues), instead of which support from people with knowledge of Portoviejo's road network was used.

5.1. QGIS and OSM plugin

QGIS was chosen because of its OSM plugin, and the possibility of building, from a GIS environment, a tool that could provide different processing options from its core modules.

This tool was developed for version 1.7 of QGIS, and is compatible with other 1.x releases. QGIS version 2 is nowadays available (it was not when the development of this tool began), and OSM processing functions are still available and have been improved. But these new capabilities are now not available as a plugin, but as a part of QGIS core functionality, making the previous plugin obsolete (OpenStreetMap Wiki, 2013). QGIS 2 does not support plugins for early QGIS versions, like the modified plugin created in this work, so it would have to be transformed to new standards and specifications to be used.

But other OSM specific developed tools still provide an easier and more powerful environment when handling OSM data, like JOSM (Java OpenStreetMap Editor), making this effort suited specially for QGIS users.

5.2. Average Speed

Average speeds are calculated from pairs of points and their coordinates. Pairs are considered valid if:

- Are less than 15 meters distant from a road
- Have the same ID
- Are not more than a minute far in time
- Beginning and ending nodes are different (not a polygon)
- Are not too close from each other

The latter criterion was necessary to consider because of the errors produced averaging speeds when a vehicle remained still for a long time on a particular location. Pairs of vehicle location coordinates, when a vehicle does not move, are considered as short movements with a minute of delay, thus misleading algorithm and the tool.

Figure 21.- Example of GPS data for a stationary vehicle

The figure above shows the distribution of GPS locations, for a bus, from a sample of data taken from 18h00 to 23h00, over a Google Earth image. This is an example of a vehicle standing still, points are roughly in a 40 meter diameter circle, but most of them inside a 16 meter diameter circle. This is a problem for a tool like the one presented here, because during extreme traffic jams a car may not move at all for a minute, or move less than 40 meters. If these groups of points are suppressed from calculations, these jams are not considered, thus calculating a wrong average speed value. But if considered, average speeds would be calculated using parking locations, flat tires events, or others. The tool should eventually consider the possibility of managing these events, and discriminate them from traffic jams.

Besides this, the tool does not discriminate GPS data by groups of hours (e.g.: morning, morning rush hours, afternoon, etc.), this causes for average speeds to merge values from rush hours and not traffic jam hours. A routing tool could use and inform this criterion to a user. But if an “hour discrimination” criterion was taken into account, then its information would have to be contained by specific tags, which should been developed and approved by communities of OSM users, before it could be used massively.

Another consideration to be made involves roads that are long, for example beginning outside the city, going through its perimeter and downtown zones, and continuing until it leaves the city again. Speed values are compulsory be higher in external zones, but if a road is represented as a unique feature, its average speed value merges all zones values. Also, if GPS data does not “cover” all road,

average speed is calculated for a portion of it, but value is assigned to the complete road.

5.3. Directions

One of the problems found when determining road directions was that assistance of a person with direction knowledge of the city roads was needed to assure accurate data. The fact that outside support is needed limits its users, as a person is needed, could not be a GIS specialist himself.

Other problem is that OSM network was not edited correctly, which was a presumption when designing the tool. Some roads should be divided in many smaller roads, but algorithm does not look for this editing mistakes. An example is exposed in the next Figure.

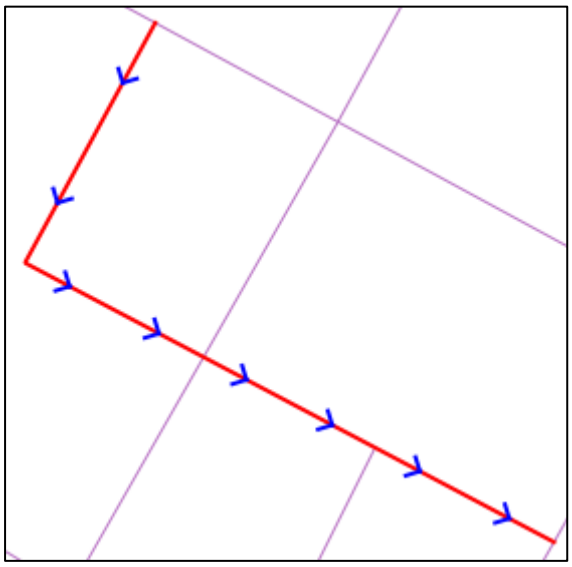


Figure 22.- Example of OSM editing issues

5.4. Performance

A laptop or a desktop computer with regular processing capabilities was used, finding performance problems when dealing with large amount of data.

When first testing was being developed, memory errors were recursive, even with small OSM datasets. This was caused by a Python resource used: python lists that were used as containers for OSM datasets and CSV datasets. This problem obligated to find an alternative, so auxiliary CSV files are created every time a process is performed, that contain all GPS and OSM data to be used when solving.

Even then, tests conducted could determine road directions and speed averages, but for a limited amount of GPS data. This could obligate to generate some strategies to process GPS data with the tool, for example:

- Use smaller GPS datasets, excerpts
- Generate datasets using vehicles schedules, looking to update a specific road or zone
- Perform a unique processing phase, and then make the corrections required as errors appear

Also, a server could be used, enhancing processing periods of time needed.

5.5. Programming errors

Testing phases 3 and 4 had to be reproduced several times until checking lists were completely positive. Final run was held without problems or issues, but any change on processing conditions (resources, environment) may need a new testing phase to solve possible problems.

6. Conclusions

- The two principal elements mentioned in this work hypothesis are average speed and road direction, expecting them to be determined with a new tool, using GPS data provided and free resources. Testing phase demonstrates that, given specific conditions, it is feasible.
- Average speed is intended to give users a routing resource so travel time is saved. This tool does calculate average speed, but due to limitations on GPS data and characteristics of the roads network of Portoviejo, its real use should still be determined.
- Portoviejo's road network, after processed with the tool developed for this work and its information about road directions being improved, is more complete than before. But, if there is no control over massive processing, it may imply massive errors. They must be performed with special attention.
- This tool should be tested with a dataset of GPS data from random vehicles, not only buses. This would allow most roads to be considered in both senses (if it is a two-way street); also average speeds would allow a more realistic average calculated.
- The tool should allow in the future other formats of GPS data, according to necessity, and according to improvements of technology.

7. References

American Public Transportation Association. (2009, january 20). *Comments to Access Board Docket Number 2007-1.* Retrieved september 08, 2013, from http://www.apta.com/gap/fedreg/documents/apta_comments_access_board_bus_2009.pdf

Asamblea Nacional Constituyente - Ecuador. (2012). *Reglamento General para la Aplicación de la Ley Orgánica de Transporte Terrestre, Tránsito y Seguridad Vial.* Quito, Ecuador.

Barret, D. J. (2008). *In O'Reilly Media, Inc.* Retrieved july 30, 2012, from Mediawiki: http://books.google.com.ec/books?id=2dhL5V3pLHkC&printsec=frontcover&hl=es#v=onepage&q&f=false

Bartlett, G. (2009, April 10). *Vehicle Tracking Systems – Which One Is Right For You?* Retrieved July 03, 2013, from http://www.rmtracking.com/blog/2009/04/10/vehicle-tracking-systems-which-one-is-right-for-you/

Chatfield, T. B. (2009). *The Complete Guide To Wikis: How to Set Up, Use, and Benefit from Wikis for Teachers, Business Professionals, Families, and Friends. (pp. 22, 25).* Retrieved July 2012, 30, from http://books.google.com.ec/books?id=TSdRIqD865sC&printsec=frontcover&hl=es#v=onepage&q&f=false

Comisión de Tránsito del Ecuador. (2011, 04 19). *Reglamento de la Ley Orgánica de Transporte Terrestre, Tránsito y Seguridad Vial.* Retrieved from http://www.cte.gob.ec/wp-content/uploads/2011/04/ReglamentoGralAplicaLOTTTSV.pdf

Corti, P. (2012, February 03). *Thinking in GIS*. Retrieved June 02, 2013, from Python for geospatial developers: http://www.paolocorti.net/2012/02/03/python_for_geospatial_developers/

Digia. (2013). *Qt |The Power of a Complete Development Framework*. Retrieved June 03, 2013, from http://qt.digia.com/Product/#.UavHe5y0TsA

Dobias, M. (2011). *PyQGIS documentation*. Retrieved June 02, 2013, from Introduction: http://www.qgis.org/pyqgis-cookbook/intro.html

Fogel, K. (2005). *Producing Open Source Software.* Retrieved 12 08, 2013, from The GPL and License Compatibility: http://producingoss.com/en/license-compatibility.html

GISLounge. (2012, November 26). *Open Source GIS and Freeware GIS Applications*. Retrieved 09 08, 2013, from http://www.gislounge.com/open-source-gis-applications/

Goodchild, M. F. (2007). Citizens as Voluntary Sensors: Spatial Data Infrastructure. *International Journal of Spatial Data Infrastructures Research, 2*, 27.

Jon, G.-B. (n.d.). *What is software licensing?* Retrieved July 30, 2012, from http://web.archive.org/web/20110721052929/http://knol.google.com/k/jon-gillespie-brown/what-is-software-licensing/3v64x901bjfe2/2#

MacWright, T. (2012, October 31). *GIS with Python, Shapely, and Fiona*. Retrieved June 02, 2013, from http://macwright.org/2012/10/31/gis-with-python-shapely-fiona.html

MapBox. (2013). *OpenStreetMap Data Report*. Retrieved august 27, 2013, from 2013: http://www.mapbox.com/osm-data-report/

Open Data Commons. (n.d.). *ODC Open Database License (ODbL) Summary*. Retrieved June 01, 2013, from http://opendatacommons.org/licenses/odbl/summary/

Open Source Geospatial Foundation. (2011). *Quantum GIS userguide.* Retrieved from version 1.7.0 'Wroclaw': http://download.osgeo.org/qgis/doc/manual/qgis-1.7.0_user_guide_en.pdf

Open Source Initiative. (n.d.). *The Open Source Definition*. Retrieved May 12, 2013, from http://opensource.org/docs/osd

OpenStreetMap. (2011, August 02). *Ecuador/normalizacion de vias*. Retrieved June 01, 2013, from http://wiki.openstreetmap.org/wiki/Ecuador/normalizacion_de_vias

OpenStreetMap. (2011, July 09). *License/We Are Changing The License*. Retrieved June 01, 2013, from http://www.osmfoundation.org/wiki/License/We_Are_Changing_The_License

OpenStreetMap. (2013, March 29). *OSM tags for routing*. Retrieved June 01, 2013, from http://wiki.openstreetmap.org/wiki/OSM_tags_for_routing

OpenStreetMap. (n.d.). *Copyright and License*. Retrieved June 01, 2013, from http://www.openstreetmap.org/copyright

OpenStreetMap Wiki. (2013). *QGIS OSM Plugin.* Retrieved december 17, 2013, from http://wiki.openstreetmap.org/wiki/QGIS_OSM_Plugin

Python Software Foundation. (2012). *The Python Tutorial*. Retrieved 2013, from 4.7.6 Documentation Strings, 4.8 Intermezzo: Coding Style: http://docs.python.org/tutorial/controlflow.html

Python Software Foundation. (2013). *About*. Retrieved June 02, 2013, from http://www.python.org/about/

Python Software Foundation. (2013). *The Python Tutorial*. Retrieved september 08, 2013, from Modules: http://docs.python.org/2/tutorial/modules.html#packages

Qt Project Hosting. (2013). *Qt Licensing*. Retrieved june 02, 2013, from http://qt-project.org/products/licensing

Quantum GIS web site. (n.d.). *QGIS Features*. Retrieved July 13, 2012, from http://www.qgis.org/en/site/about/features.html

Riverbank Computing Limited. (2011). *PyQt 4.10.1 Reference Guide*. Retrieved June 2013, 2013, from Introduction: http://pyqt.sourceforge.net/Docs/PyQt4/introduction.html#license

Riverbank Computing Limited. (2013). *PyQt*. Retrieved June 02, 2013, from License: http://www.riverbankcomputing.co.uk/software/pyqt/license

Sherman, G. (2011, December 19). *QGIS users around the world.* Retrieved July 15, 2012, from Blog Post: http://spatialgalaxy.net/2011/12/19/qgis-users-around-the-world/

The Quantum GIS project. (n.d.). *About QGIS*. Retrieved June 02, 2013, from http://www.qgis.org/en/about-qgis.html

Tomer, C. (2002). *"Open Source." Computer Sciences.* Retrieved July 26, 2012, from http://www.encyclopedia.com/doc/1G2-3401200413.html

Universidad Carlos III de Madrid. (2013). *Oficina de Información Científica - Noticias*. Retrieved september 08, 2013, from A system that improves the precision of GPS in cities by 90 percent: http://www.uc3m.es/portal/page/portal/actualidad_cientifica/noticias/improves_precision_gps

Westra, E. (2010). *Python Geospatial Development.* Birmingham: Packt Publishing.

Wikibooks. (2013, May 16). *Python Programming*. Retrieved June 02, 2013, from http://en.wikibooks.org/wiki/Python_Programming

Wikibooks. (2013, January 19). *Python Programming/Overview*. Retrieved June 02, 2013, from http://en.wikibooks.org/wiki/Python_Programming/Overview

World Intellectual Property Organization. (2004). *The Concept of Intellectual Property.* Retrieved July 26, 2012, from WIPO Intellectual Property Handbook. Second Edition: http://www.wipo.int/export/sites/www/about-ip/en/iprm/pdf/ip_handbook.pdf

World Wide Web Consortium. (2012). *Extensible Markup Language (XML)*. Retrieved May 2013, 31, from http://www.w3.org/XML/

ZetCode. (2012, September 26). *Introduction to Qt4 toolkit*. Retrieved June 02, 2013, from http://www.zetcode.com/gui/qt4/introduction/

8. Appendices

Appendix 1: Testing phase Check List

#	Topic	Questions	One way tag			
			"Yes"	"-1"	"No"	No tag
1	Maximum speed	Is it added when commanded?				
2	Average speed	Is it accurate?				
3		Is it added when commanded?				
4	Road direction	Is it accurate?				
5		Does it respond to logical flow?				
6	Procedure errors	Detected errors were corrected?				

Table 13.- Checklist used in Testing phase

Appendix 2: Python Syntax

The Python Tutorial (Python Software Foundation, 2012) is available online and proposes some python writing guidelines that ensure readability in Python Scripts. Highlights of this coding style are mentioned below.

- Wrap lines so that they don't exceed 79 characters.
- Use blank lines to separate functions and classes, and larger blocks of code inside functions.
- When possible, put comments on a line of their own.
- Use spaces around operators and after commas, but not directly inside bracketing constructs: `a = f(1, 2) + g(3, 4).`
- Name your classes and functions consistently; the convention is to use `CamelCase` for classes and `lower_case_with_underscores` for functions and methods. Always use `self` as the name for the first method argument
- Don't use fancy encodings if your code is meant to be used in international environments. Plain ASCII works best in any case.

Besides these above mentioned items, classes were given an explanatory text to be read when help function is used.